

Nonlinear Dimensionality Reduction with Side Information

by

Ali Ghodsi

A thesis

presented to the University of Waterloo

in fulfillment of the

thesis requirement for the degree of

Doctor of Philosophy

in

Computer Science

Waterloo, Ontario, Canada, 2005

©Ali Ghodsi, 2005

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A THESIS

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners. I understand that my thesis may be made electronically available to the public.

Abstract

In this thesis, I look at three problems with important applications in data processing. Incorporating side information, provided by the user or derived from data, is a main theme of each of these problems.

This thesis makes a number of contributions. The first is a technique for combining different embedding objectives, which is then exploited to incorporate side information expressed in terms of transformation invariants known to hold in the data. It also introduces two different ways of incorporating transformation invariants in order to make new similarity measures. Two algorithms are proposed which learn metrics based on different types of side information. These learned metrics can then be used in subsequent embedding methods. Finally, it introduces a manifold learning algorithm that is useful when applied to sequential decision problems. In this case we are given action labels in addition to data points. Actions in the manifold learned by this algorithm have meaningful representations in that they are represented as simple transformations.

To my mother and the memory of my father

Acknowledgements

I owe thanks to many people for the completion of this thesis:

- To Dale Schuurmans who has been a very good friend, an excellent advisor and with whom it has been a pleasure and privilege to work. I would like to thank him for his excellent advice and free donation of his time. My research was greatly improved by the high standards with which he conducts his own work.
- To Brendan Frey for the opportunities he afforded me and for his incisive advice.
- To Michael Bowling who has been a stimulating mentor as well as a friend.
- To Finnegan Southey, my constant friend, for being an excellent collaborator. My research was greatly improved by his suggestions and his tremendous help.
- To Richard Mann, Mu Zhu, and Adam Krzyzak, for kindly agreeing to act as examiners. Not only was their feedback insightful, but they were remarkably kind. From their feedback I benefited greatly.
- To Sam Roweis and Peter Van Beek, for their guidance.
- To Paul Nijjar , Dana Wilkinson, Jiayuan Huang, and Shojaeddin Chenouri for the productive discussions we shared and for always being willing to engage in last minute problem solving sessions.

- To my schoolmates, colleagues, and friends for their comments, suggestions, criticism, and fun: Masoud, Shaojun , Relu, Chris, Linli, Feng, Fuchun, Fletcher, Steven, Romer, Trausti, Anitha, Kannan, and Adam.
- To my family particularly my sister Sara who influenced my life greatly, and especially to my parents for, well, everything.
- To my son Soroush for his love, encouragement with his childish words and his sweet smiles.
- To my wife Mahtab for her optimism, and resolve. This thesis is as much a product of her support, and patience as it is of my efforts. Thank you Mahtab.

Contents

1	Introduction	1
1.1	Principal Components Analysis	5
1.1.1	Dual PCA	9
1.2	Kernel PCA	12
1.2.1	Centering	15
1.3	Locally Linear Embedding	16
1.4	Laplacian Eigenmaps	19
1.5	Metric Multidimensional Scaling (MDS)	20
1.6	Isomap	23
1.7	Semidefinite Embedding (SDE)	25
1.8	Unified Framework	27
1.9	Thesis Contributions	29

2	Hybrid Embedding	34
2.1	Introduction	34
2.2	Combined Embedding	36
2.3	Illustration: Combining Isomap and LLE	40
2.4	Incorporating Transformation Invariants as Side Information	42
2.4.1	Transformations	43
2.5	Using Tangent Correction Distance	45
2.6	Transformation Reconstruction Distance	49
2.7	Results	50
2.8	Conclusions	57
3	Learning a Metric for Embedding	60
3.1	Introduction	60
3.2	Learning a Metric from Class-Equivalence Side Information	62
3.2.1	Derivation	63
3.2.2	Results	67
3.3	Multiple-Attribute Metric Learning with Class-Equivalence Side Information	78
3.3.1	Results	80
3.4	Kernelizing Metric Learning	81
3.5	Using Partial Distance Side Information	84

3.5.1	Results	87
3.6	Conclusions	93
4	Action Respecting Embedding	95
4.1	Introduction	95
4.2	Action Respecting Embedding	98
4.2.1	Non-Uniform Neighbourhoods	99
4.2.2	Action Respecting Constraints	101
4.2.3	Data Prediction	104
4.3	Results	106
4.3.1	IMAGEBOT Domain	107
4.3.2	Manifold Learning	110
4.3.3	Data Prediction	114
4.4	Conclusions	116
5	Conclusion	118
A		123

List of Tables

1.1	<i>Direct PCA Algorithm</i>	10
1.2	<i>Dual PCA Algorithm</i>	12
1.3	<i>SDE Algorithm</i>	27
4.1	ARE Algorithm.	104
4.2	Prediction accuracy across the four trajectories.	116

List of Figures

1.1	A canonical dimensionality reduction problem	2
1.2	PCA	6
1.3	Kernel PCA	13
1.4	LLE	17
1.5	LEM	21
1.6	MDS	22
1.7	Isomap	24
1.8	SDE	28
2.1	Swiss roll	41
2.2	Hybrids embedding	42
2.3	<i>Illustration of local pixel transformation to produce x' from x.</i>	44
2.4	<i>Data and tangents along the manifold</i>	46

2.5	<i>Alignment of tangent vectors</i>	48
2.6	Three digits, tangent correction distance	52
2.7	Ten digits with tangent correction distance, and effect of α	53
2.8	<i>Blowup of sections of TC-LEM results from Figure 2.7.</i>	54
2.9	<i>Blowup of sections of TC-LEM results from Figure 2.7.</i>	54
2.10	<i>Original digit and teapot images used in experiments.</i>	55
2.11	Ten digits, tangent correction distance	55
2.12	Ten digits, transformation reconstruction distance	56
2.13	Teapot viewed from two different angles, tangent correction distance	58
2.14	Teapot viewed from two different angles, transformation reconstruction distance	58
3.1	Equivalence-Informed MDS (bearded/unbearded, all pairs)	69
3.2	Equivalence-Informed MDS (bearded/unbearded)	70
3.3	Equivalence-Informed Isomap (bearded/unbearded, all pairs)	71
3.4	Equivalence-Informed Isomap (bearded/unbearded)	72
3.5	Equivalence-Informed LEM (bearded/unbearded, all pairs)	73
3.6	Equivalence-Informed LEM (bearded/unbearded)	74
3.7	Equivalence-Informed LLE (bearded/unbearded, all pairs)	75
3.8	Equivalence-Informed LLE (bearded/unbearded)	76
3.9	Equivalence-Informed SDE (bearded/unbearded, all pairs)	77

3.10	Equivalence-Informed SDE (bearded/unbearded)	79
3.11	Multi-Attribute Equivalence-Informed MDS	82
3.12	Distance-Informed MDS (bearded/unbearded)	89
3.13	Distance-Informed Isomap (bearded/unbearded)	90
3.14	Distance-Informed LEM (bearded/unbearded)	91
3.15	Distance-Informed LLE (bearded/unbearded)	92
3.16	Distance-Informed SDE (bearded/unbearded)	94
4.1	non-uniform neighbourhoods	100
4.2	IMAGEBOT's world.	107
4.3	<i>A sample 60-action trajectory from IMAGEBOT.</i>	108
4.4	A more complicated 45-action trajectory from IMAGEBOT	109
4.5	Manifolds from trajectory shown in Figure 4.3	111
4.6	Manifolds from trajectory shown in Figure 4.3 with different actions	112
4.7	<i>Manifolds learned on data generated with rotation actions.</i>	113
4.8	<i>Manifolds corresponding to Figure 4.4</i>	114
4.9	<i>Manifolds learned on data generated with zoom actions.</i>	115
A.1	Equivalence-Informed MDS (glasses/no glasses), all pairs	124
A.2	Equivalence-Informed MDS (glasses/no glasses)	125

A.3	Equivalence-Informed Isomap (glasses/no glasses), all pairs	126
A.4	Equivalence-Informed Isomap (glasses/no glasses)	127
A.5	Equivalence-Informed LEM (glasses/no glasses), all pairs	128
A.6	Equivalence-Informed LEM (glasses/no glasses)	129
A.7	Equivalence-Informed LLE (glasses/no glasses), all pairs	130
A.8	Equivalence-Informed LLE (glasses/no glasses)	131
A.9	Equivalence-Informed SDE (glasses/no glasses), all pairs	132
A.10	Equivalence-Informed SDE (glasses/no glasses)	133
A.11	Multiple-Attribute Equivalence-Informed Isomap	134
A.12	Multiple-Attribute Equivalence-Informed LEM	135
A.13	Multiple-Attribute Equivalence-Informed LLE	136
A.14	Multiple-Attribute Equivalence-Informed SDE	137
A.15	Distance-Informed MDS (glasses/no glasses)	138
A.16	Distance-Informed Isomap (glasses/no glasses)	139
A.17	Distance-Informed LEM (glasses/no glasses)	140
A.18	Distance-Informed LLE (glasses/no glasses)	141
A.19	Distance-Informed SDE (glasses/no glasses)	142

Chapter 1

Introduction

Manifold learning is a significant problem across a wide variety of information processing fields including pattern recognition, data compression, machine learning, and database navigation. In many problems, the measured data vectors are high-dimensional but we may have reason to believe that the data lie near a lower-dimensional manifold. In other words, we may believe that high-dimensional data are multiple, indirect measurements of an underlying source, which typically cannot be directly measured. Learning a suitable low-dimensional manifold from high-dimensional data is essentially the same as learning this underlying source.

Dimensionality reduction¹ can also be seen as the process of deriving a set of degrees

¹ In this thesis ‘manifold learning’ and ‘dimensionality reduction’ are used interchangeably.

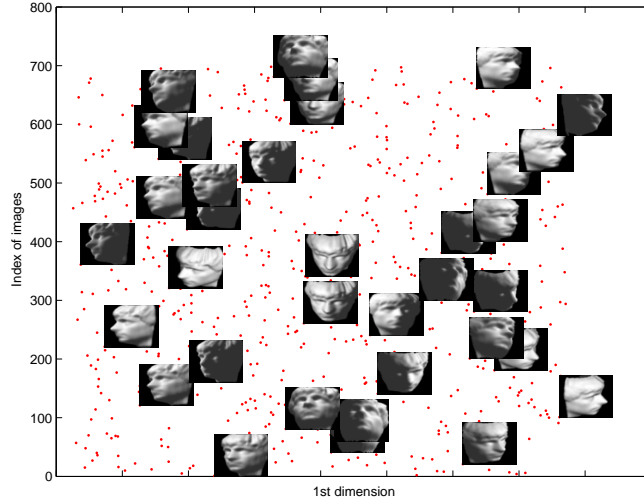


Figure 1.1: *A canonical dimensionality reduction problem from visual perception. The input consists of a sequence of 4096-dimensional vectors, representing the brightness values of 64 pixel by 64 pixel images of a face. Applied to $N = 698$ raw images. The first coordinate axis of the embedding correlates highly with one of the degrees of freedom underlying the original data: left-right pose.*

of freedom which can be used to reproduce most of the variability of a data set. Consider a set of images produced by the rotation of a face through different angles. Clearly only one degree of freedom is being altered, and thus the images lie along a continuous one-dimensional curve through image space. Figure 1.1 shows an example of image data that exhibits one intrinsic dimension.

Manifold learning techniques can be used in different ways including:

- Data dimensionality reduction: Produce a compact low-dimensional encoding of a given high-dimensional data set.
- Data visualization: Provide an interpretation of a given data set in terms of intrinsic degree of freedom, usually as a by-product of data dimensionality reduction.
- Preprocessing for supervised learning: Simplify, reduce, and clean the data for subsequent supervised training.

Many algorithms for dimensionality reduction have been developed to accomplish these tasks. However, since the need for such analysis arises in many areas of study, contributions to the field have come from many disciplines. While all of these methods have a similar goal, approaches to the problem are different.

Principal components analysis (PCA) [18] is a classical method that provides a sequence of best linear approximations to a given high-dimensional observation. It is one of the most popular techniques for dimensionality reduction. However, its effectiveness is limited by its global linearity. Multidimensional scaling (MDS) [9], which is closely related to PCA, suffers from the same drawback. Factor analysis [11, 30] and independent component analysis (ICA) [17] also assume that the underlying manifold is a linear subspace. However, they differ from PCA in the way they identify and model the subspace. The subspace modeled by PCA captures the maximum variability in the data, and can be viewed as

modeling the covariance structure of the data, whereas factor analysis models the correlation structure. ICA starts from a factor analysis solution and searches for rotations that lead to independent components [30, 7]. The main drawback with all these classical dimensionality reduction approaches is that they only characterize *linear* subspaces (manifolds) in the data. In order to resolve the problem of dimensionality reduction in *nonlinear* cases, many recent techniques, including kernel PCA [21, 26], locally linear embedding (LLE) [23, 24], Laplacian eigenmaps (LEM) [1], Isomap [31, 32], and semidefinite embedding (SDE) [35, 34] have been proposed.

In this thesis I present many extensions of modern nonlinear dimensionality reduction techniques based on using additional side information to obtain better low-dimensional embeddings in natural data sets. The remainder of this chapter provides a brief overview of different existing approaches and shows their close connection, and sets the stage for the specific contributions I make in this thesis. Section 2 of this chapter explains principal components analysis, which is the core of many other techniques. Then, in sections 3, 4 and 5 I discuss recent nonlinear extensions to PCA that have been proposed, including kernel PCA, locally linear embedding, and Laplacian eigenmaps respectively. Multidimensional scaling and its recent nonlinear extension, Isomap, are then discussed in Sections 6 and 7. Section 8 discusses semidefinite embedding, a new approach to dimensionality reduction based on semidefinite programming. Finally, a unified framework that represents all of

these techniques as different variations of kernel PCA is then introduced in Section 9. I conclude the chapter with a statement of the main contributions of this thesis.

1.1 Principal Components Analysis

Principal components analysis (PCA) is a very popular technique for dimensionality reduction. Given a set of data on n dimensions, PCA aims to find a linear subspace of dimension d lower than n such that the data points lie mainly on this linear subspace (See Figure 1.2 as an example of a two-dimensional projection found by PCA). Such a reduced subspace attempts to maintain most of the variability of the data.

The linear subspace can be specified by d orthogonal vectors that form a new coordinate system, called the ‘principal components’. The principal components are orthogonal, linear transformations of the original data points, so there can be no more than n of them. However, the hope is that only $d < n$ principal components are needed to approximate the space spanned by the n original axes.

The most common definition of PCA, due to Hotelling [16], is that, for a given set of data vectors x_i , $i \in 1...t$, the d principal axes are those orthonormal axes onto which the variance retained under projection is maximal.

In order to capture as much of the variability as possible, let us choose the first prin-

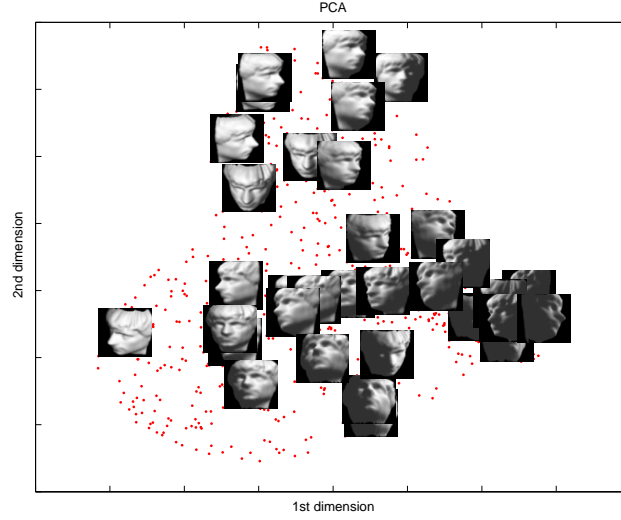


Figure 1.2: *PCA applied to the same data set. A two-dimensional projection is shown, with a sample of the original input images.*

principal component, denoted by U_1 , to have maximum variance. Suppose that all centered observations are stacked into the columns of an $n \times t$ matrix X , where each column corresponds to an n -dimensional observation and there are t observations. Let the first principal component be a linear combination of X defined by coefficients (or weights) $w = [w_1 \dots w_n]$.

In matrix form:

$$U_1 = w^T X$$

$$\text{var}(U_1) = \text{var}(w^T X) = w^T S w$$

where S is the $n \times n$ sample covariance matrix of X .

Clearly $\text{var}(U_1)$ can be made arbitrarily large by increasing the magnitude of w . Therefore, we choose w to maximize $w^T Sw$ while constraining w to have unit length.

$$\begin{aligned} \max \quad & w^T Sw \\ \text{subject to} \quad & w^T w = 1 \end{aligned}$$

To solve this optimization problem a Lagrange multiplier α_1 is introduced:

$$L(w, \alpha) = w^T Sw - \alpha_1(w^T w - 1) \quad (1.1)$$

Differentiating with respect to w gives n equations,

$$Sw = \alpha_1 w$$

Premultiplying both sides by w^T we have:

$$w^T Sw = \alpha_1 w^T w = \alpha_1$$

$\text{var}(U_1)$ is maximized if α_1 is the largest eigenvalue of S .

Clearly α_1 and w are an eigenvalue and an eigenvector of S . Differentiating (1.1) with respect to the Lagrange multiplier α_1 gives us back the constraint:

$$w^T w = 1$$

This shows that the first principal component is given by the normalized eigenvector with the largest associated eigenvalue of the sample covariance matrix S . A similar ar-

gument can show that the d dominant eigenvectors of covariance matrix S determine the first d principal components.

Another nice property of PCA, closely related to the original discussion by Pearson [22], is that the projection onto the principal subspace minimizes the squared reconstruction error, $\sum_{i=1}^t ||x_i - \hat{x}_i||^2$. In other words, the principal components of a set of data in \Re^n provide a sequence of best linear approximations to that data, for all ranks $d \leq n$.

Consider the rank- d linear approximation model as :

$$f(y) = \bar{x} + U_d y$$

This is the parametric representation of a hyperplane of rank d .

For convenience, suppose $\bar{x} = 0$ (otherwise the observations can be simply replaced by their centered versions $\tilde{x} = x_i - \bar{x}$). Under this assumption the rank d linear model would be $f(y) = U_d y$, where U_d is a $n \times d$ matrix with d orthogonal unit vectors as columns and y is a vector of parameters. Fitting this model to the data by least squares leaves us to minimize the reconstruction error:

$$\min_{U_d, y_i} \sum_i^t ||x_i - U_d y_i||^2$$

By partial optimization for y_i we obtain:

$$\frac{d}{dy_i} = 0 \Rightarrow y_i = U_d^T x_i$$

Now we need to find the orthogonal matrix U_d :

$$\min_{U_d} \sum_i^t ||x_i - U_d U_d^T x_i||^2$$

Define $H_d = U_d U_d^T$. H_d is a $n \times n$ matrix which acts as a projection matrix and projects each data point x_i onto its rank d reconstruction. In other words, $H_d x_i$ is the orthogonal projection of x_i onto the subspace spanned by the columns of U_d . A unique solution U can be obtained by finding the singular value decomposition of X [30]. For each rank d , U_d consists of the first d columns of U .

Clearly the solution for U can be expressed as singular value decomposition (SVD) of X .

$$X = U \Sigma V^T$$

since the columns of U in the SVD contain the eigenvectors of XX^T . The PCA procedure is summarized in Algorithm 1 (see Table 1.1).

1.1.1 Dual PCA

It turns out that the singular value decomposition also allows us to formulate the principle components algorithm entirely in terms of dot products between data points and limit the direct dependence on the original dimensionality n . This fact will become important below.

Algorithm 1

Recover basis: Calculate $XX^\top = \sum_{i=1}^t x_i x_i^\top$ and let U = eigenvectors of XX^\top corresponding to the top d eigenvalues.

Encode training data: $Y = U^\top X$ where Y is a $d \times t$ matrix of encodings of the original data.

Reconstruct training data: $\hat{X} = UY = UU^\top X$.

Encode test example: $y = U^\top x$ where y is a d -dimensional encoding of x .

Reconstruct test example: $\hat{x} = Uy = UU^\top x$.

Table 1.1: *Direct PCA Algorithm*

Assume that the dimensionality n of the $n \times t$ matrix of data X is large (i.e., $n \gg t$). In this case, Algorithm 1 (Table 1.1) is impractical. We would prefer a run time that depends only on the number of training examples t , or that at least has a reduced dependence on n .

Note that in the SVD factorization $X = U\Sigma V^T$, the eigenvectors in U corresponding to nonzero singular values in Σ (square roots of eigenvalues) are in a one-to-one correspon-

dence with the eigenvectors in V .

Now assume that we perform dimensionality reduction on U and keep only the first d eigenvectors, corresponding to the top d nonzero singular values in Σ . These eigenvectors will still be in a one-to-one correspondence with the first d eigenvectors in V :

$$X V = U \Sigma$$

where the dimensions of these matrices are:

$$\begin{array}{cccc} X & U & \Sigma & V \\ n \times t & n \times d & d \times d & t \times d \\ & & \text{diagonal} & \end{array}$$

Crucially, Σ is now square and invertible, because its diagonal has nonzero entries. Thus, the following conversion between the top d eigenvectors can be derived:

$$U = X V \Sigma^{-1} \tag{1.2}$$

Replacing all uses of U in Algorithm 1 with $XV\Sigma^{-1}$ gives us the dual form of PCA, Algorithm 2 (see Table 1.2). Note that in Algorithm 2 (Table 1.2), the steps of “*Reconstruct training data*” and “*Reconstruction test example*” still depend on n , and therefore still will be impractical in the case that the original dimensionality n is very large. However all other steps can be done conveniently in the run time that depends only on the number of training examples t .

Algorithm 2

Recover basis: Calculate $X^\top X$ and let V = eigenvectors of $X^\top X$ corresponding to the top d eigenvalues. Let Σ = diagonal matrix of *square roots* of the top d eigenvalues.

Encode training data: $Y = U^\top X = \Sigma V^\top$ where Y is a $d \times t$ matrix of encodings of the original data.

Reconstruct training data: $\hat{X} = UY = U\Sigma V^\top = XV\Sigma^{-1}\Sigma V^\top = XVV^\top$.

Encode test example: $y = U^\top x = \Sigma^{-1}V^\top X^\top x = \Sigma^{-1}V^\top X^\top x$ where y is a d dimensional encoding of x .

Reconstruct test example: $\hat{x} = Uy = UU^\top x = XV\Sigma^{-2}V^\top X^\top x = XV\Sigma^{-2}V^\top X^\top x$.

Table 1.2: *Dual PCA Algorithm*

1.2 Kernel PCA

PCA is designed to model linear variabilities in high-dimensional data. However, many high dimensional data sets have a nonlinear nature. In these cases the high-dimensional data lie

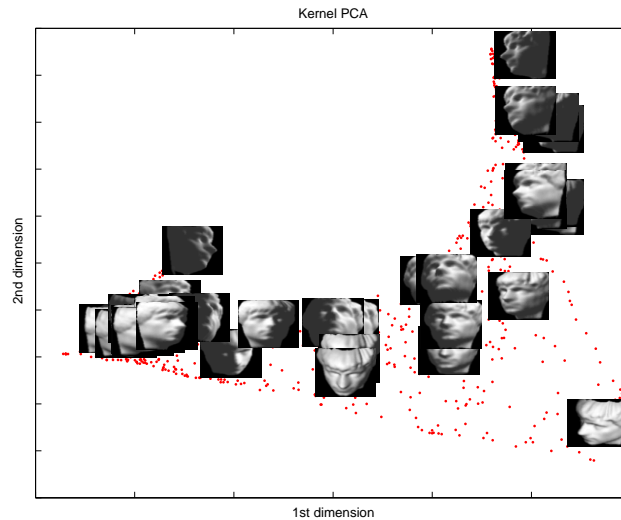


Figure 1.3: *Kernel PCA with Gaussian kernel applied to the same data set. A two-dimensional projection is shown, with a sample of the original input images.*

on or near a nonlinear manifold (not a linear subspace) and therefore PCA can not model the variability of the data correctly. One of the algorithms designed to address the problem of nonlinear dimensionality reduction is Kernel PCA (See Figure 1.3 for an example). In Kernel PCA, through the use of kernels, principle components can be computed efficiently in high-dimensional feature spaces that are related to the input space by some nonlinear mapping.

Kernel PCA finds principal components which are nonlinearly related to the input space by performing PCA in the space produced by the nonlinear mapping, where the low-dimensional latent structure is, hopefully, easier to discover.

Consider a feature space \mathcal{H} such that:

$$\Phi : x \rightarrow \mathcal{H}$$

$$x \mapsto \Phi(x)$$

Suppose $\sum_i^t \Phi(x_i) = 0$ (we will return to this point below, and show how this condition can be satisfied in a Hilbert space). This allows us to formulate the kernel PCA objective as follows:

$$\min \sum_i^t ||\Phi(x_i) - U_q U_q^T \Phi(x_i)||$$

By the same argument used for PCA, the solution can be found by SVD:

$$\Phi(X) = U \Sigma V^T$$

where U contains the eigenvectors of $\Phi(X)\Phi(X)^T$. Note that if $\Phi(X)$ is $\mathbf{n} \times t$ and the dimensionality of the feature space \mathbf{n} is large, then U is $\mathbf{n} \times \mathbf{n}$ which will make PCA impractical.

To reduce the dependence on \mathbf{n} , first assume that we have a kernel $K(\cdot, \cdot)$ that allows us to compute $K(x, y) = \Phi(x)^\top \Phi(y)$. Given such a function, we can then compute the matrix $\Phi(X)^\top \Phi(X) = K$ efficiently, without computing $\Phi(X)$ explicitly. Crucially, K is $t \times t$ here and does not depend on \mathbf{n} . Therefore it can be computed in a run time that depends only on t . Also, note that PCA can be formulated entirely in terms of dot products between

data points (Algorithm 2 represented in Table 1.2). Replacing dot products in Algorithm 2 (1.2) by kernel function K , which is in fact equivalent to the inner product of a Hilbert space yields to the Kernel PCA algorithm.

1.2.1 Centering

In the derivation of the kernel PCA we assumed that $\Phi(X)$ has zero mean. The following normalization of the kernel satisfies this condition.

$$\tilde{K}(x, y) = K(x, y) - E_x[K(x, y)] - E_y[K(x, y)] + E_x[E_y[K(x, y)]]$$

In order to prove that, define:

$$\tilde{\Phi}(X) = \Phi(X) - E_x[\Phi(X)]$$

Finally, the corresponding kernel is:

$$\tilde{K}(x, y) = \tilde{\Phi}(x)\tilde{\Phi}(y)$$

This expands as follows:

$$\begin{aligned}\tilde{K}(x, y) &= (\Phi(x) - E_x[\Phi(x)]).(\Phi(y) - E_y[\Phi(y)]) \\ &= K(x, y) - E_x[K(x, y)] - E_y[K(x, y)] + E_x[E_y[K(x, y)]]\end{aligned}$$

To perform Kernel PCA, one needs to replace all dot products $x^T y$ by $\tilde{K}(x, y)$ in Algorithm 2 (Table 1.2). Note that V is the eigenvectors of $K(X, X)$ corresponding to the top d eigenvalues, and Σ is diagonal matrix of square roots of the top d eigenvalues.

Unfortunately Kernel PCA does not inherit all the strength of PCA. More specifically reconstruction of training and test data points is not a trivial practice in Kernel PCA. Algorithm 2 (Table 1.2) shows that data can be reconstructed in feature space $\hat{\Phi}(x)$. However finding the corresponding pattern x is difficult and sometimes even impossible [25].

1.3 Locally Linear Embedding

Locally linear embedding (LLE) is another approach which address the problem of nonlinear dimensionality reduction (See Figure 1.4 for an example) by computing low-dimensional, neighbourhood preserving embedding of high-dimensional data. A data set of dimensionality n , which is assumed to lie on or near a smooth nonlinear manifold of dimensionality $d < n$, is mapped into a single global coordinate system of lower dimensionality, d . The global nonlinear structure is recovered by locally linear fits.

Consider t n -dimensional real-valued vectors x_i sampled from some underlying manifold. We can assume each data point and its neighbours lie on, or are close to, a locally linear

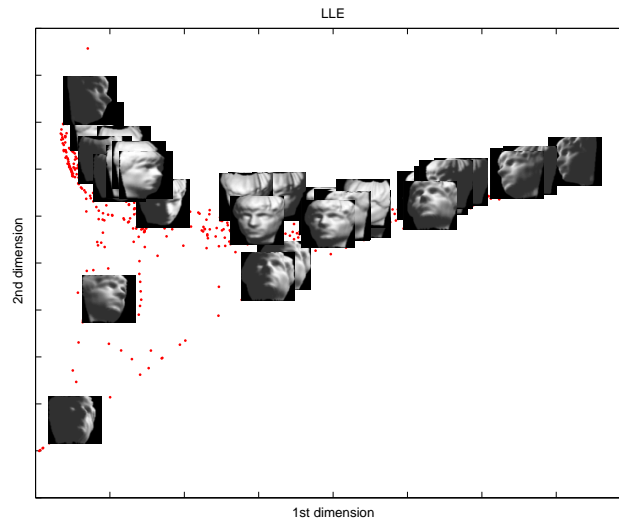


Figure 1.4: *LLE applied ($k = 5$) to the same data set. A two-dimensional projection is shown, with a sample of the original input images.*

patch of the manifold. By a linear mapping, consisting of a translation, rotation, and rescaling, the high-dimensional coordinates of each neighbourhood can be mapped to global internal coordinates on the manifold. Thus, the nonlinear structure of the data can be identified through two linear steps: first, compute the locally linear patches, and second, compute the linear mapping to the coordinate system on the manifold.

The main goal here is to map the high-dimensional data points to the single global coordinate system of the manifold such that the relationships between neighbouring points are preserved. This proceeds in three steps:

1. Identify the neighbours of each data point x_i . This can be done by finding the k nearest neighbours, or by choosing all points within some fixed radius, ϵ .
2. Compute the weights that best linearly reconstruct x_i from its neighbours.
3. Find the low-dimensional embedding vector y_i which is best reconstructed by the weights determined in the previous step.

After finding the nearest neighbours in the first step, the second step must compute a local geometry for each locally linear patch. This geometry is characterized by linear coefficients that reconstruct each data point from its neighbours.

$$\min_w \sum_{i=1}^t \left\| \mathbf{x}_i - \sum_{j=1}^k w_{ij} \mathbf{x}_{N_i(j)} \right\|^2$$

where $N_i(j)$ is the index of the j th neighbour of the i th point. It then selects code vectors so as to preserve the reconstruction weights by solving

$$\min_Y \sum_{i=1}^t \left\| \mathbf{y}_i - \sum_{j=1}^k w_{ij} \mathbf{y}_{N_i(j)} \right\|^2$$

This objective can be reformulated as

$$\min_Y \text{Tr}(Y^T Y L) \tag{1.3}$$

where $L = (I - W)^T(I - W)$.

The solution for Y can have an arbitrary origin and orientation. In order to make the problem well-posed, these two degrees of freedom must be removed. Requiring the

coordinates to be centered on the origin ($\sum_i y_i = 0$), and constraining the embedding vectors to have unit covariance ($Y^T Y = I$), removes the first and second degrees of freedom respectively.

The cost function can be optimized initially by the second of these two constraints. Under this constraint, the cost is minimized when the columns of Y^T (rows of Y) are the eigenvectors associated with the lowest eigenvalues of L .

Discarding the eigenvector associated with eigenvalue 0 satisfies the first constraint.

1.4 Laplacian Eigenmaps

A closely related approach to locally linear embedding is Laplacian eigenmaps (See Figure 1.5 for an example). Given t points in n -dimensional space, the Laplacian eigenmaps Method (LEM) [1] starts by constructing a weighted graph with t nodes and a set of edges connecting neighbouring points. Similar to LLE, the neighbourhood graph can be constructed by finding the k nearest neighbours, or by choosing all points within some fixed radius ϵ . For weighting the edges, there are two variations: either each edge is weighted by $W_{ij} = e^{-\frac{\|x_i - x_j\|^2}{s}}$, where s is a free parameter which should be chosen a priori, or simply all W_{ij} is set to 1 if vertices i and j are connected. The embedding map is then provided

by the following objective

$$\min_Y \sum_{i=1}^t \sum_{j=1}^t (\mathbf{y}_i - \mathbf{y}_j)^2 W_{ij}$$

subject to appropriate constraints. This objective can be reformulated as

$$\min_Y \text{Tr}(YLY^T)$$

where $L = R - W$, R is diagonal, and $R_{ii} = \sum_{j=1}^t W_{ij}$. This L is called the *Laplacian function*. Similar to (1.3), after adding orthogonality and centering constraint, a solution to this problem can be found by making Y to be the eigenvectors of L (non-normalized solution). As an alternative, (1.3) can be constrained to $Y^T LY = I$. In this case, the solution is provided by the eigenvectors of the generalized eigenvalue problem $My = \lambda Dy$ (normalized solution). Note that the final objectives for both LEM and LLE have the same form and differ only in how the matrix L is constructed. Therefore, same closed form solution (taking Y to be the eigenvectors of L) works.

1.5 Metric Multidimensional Scaling (MDS)

An alternative perspective on dimensionality reduction is offered by Multidimensional scaling (MDS). MDS is another classical approach that maps the original high dimensional space to a lower dimensional space, but does so in an attempt to preserve pairwise distances (See Figure 1.6 for an example). That is MDS addresses the problem of constructing

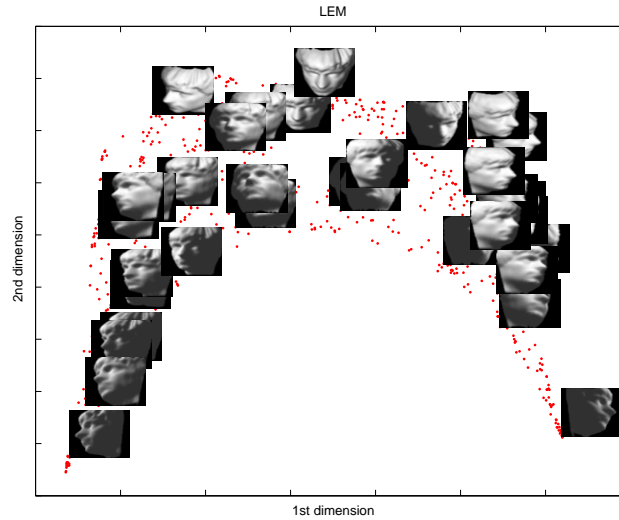


Figure 1.5: *LEM applied ($k = 7$) to the same data set. A two-dimensional projection is shown, with a sample of the original input images.*

a configuration of t points in Euclidean space by using information about the distances between the t patterns. Although it has a very different mathematics from PCA, it winds up being closely related, and in fact yields a linear embedding, as we will see.

A $t \times t$ matrix D is called a distance or affinity matrix if it is symmetric, $d_{ii} = 0$, and $d_{ij} > 0$, $i \neq j$.

Given a distance matrix D , MDS attempts to find t data points y_1, \dots, y_t in d dimensions, such that if \hat{d}_{ij} denotes the Euclidean distance between y_i and y_j , then \hat{D} is similar to D .

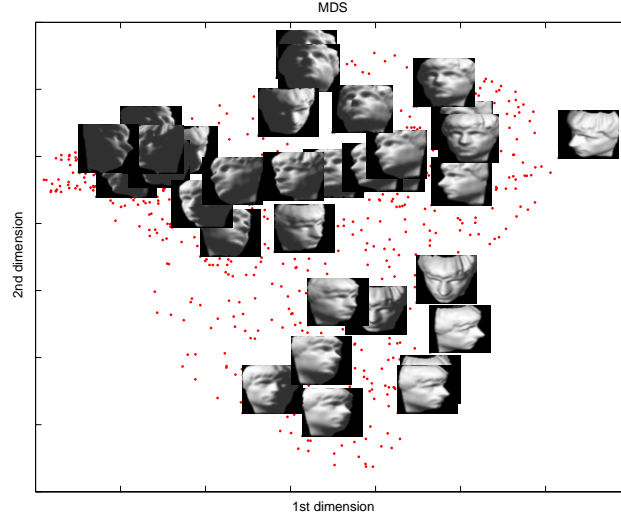


Figure 1.6: *MDS applied to the same data set. A two-dimensional projection is shown, with a sample of the original input images.*

In particular, we consider metric MDS [9], which minimizes

$$\min_Y \sum_{i=1}^t \sum_{j=1}^t (\mathbf{d}_{ij}^{(X)} - \mathbf{d}_{ij}^{(Y)})^2 \quad (1.4)$$

where $\mathbf{d}_{ij}^{(X)} = \|x_i - x_j\|^2$ and $\mathbf{d}_{ij}^{(Y)} = \|y_i - y_j\|^2$. The distance matrix $D^{(X)}$ can be converted to a kernel matrix of inner products $X^T X$ by

$$X^T X = -\frac{1}{2} H D^{(X)} H$$

where $H = I - \frac{1}{t} e e^T$ and e is a column vector of all 1's. Now (1.4) can be reduced to

$$\min_Y \sum_{i=1}^t \sum_{j=1}^t (x_i^T x_j - y_i^T y_j)^2$$

It can be shown [9] that the solution is $Y = \Lambda^{1/2}V^T$ where V is the eigenvectors of X^TX corresponding to the top d eigenvalues, and Λ is the top d eigenvalues of X^TX . Clearly the solution for MDS is identical to dual PCA (see Table 1.2), and as far as Euclidean distance is concerned, MDS and PCA produce the same results. However, the distances need not be based on Euclidean distances and can represent many types of dissimilarities between objects.

1.6 Isomap

Similar to PCA, MDS has been recently extended to perform nonlinear dimensionality reduction. A recent approach to nonlinear dimensionality reduction based on MDS is the Isomap algorithm (See Figure 1.7 for an example). Unlike the linear case, nonlinear forms of MDS are different from nonlinear forms of PCA—a fact I exploit in Chapter 2 below.

Isomap is a nonlinear generalization of classical MDS. The main idea is to perform MDS, not in the input space, but in the geodesic space of the nonlinear data manifold. The geodesic distances represent the shortest paths along the curved surface of the manifold measured as if the surface were flat. This can be approximated by a sequence of short steps between neighbouring sample points. Isomap then applies MDS to the geodesic rather than straight line distances to find a low-dimensional mapping that preserves these

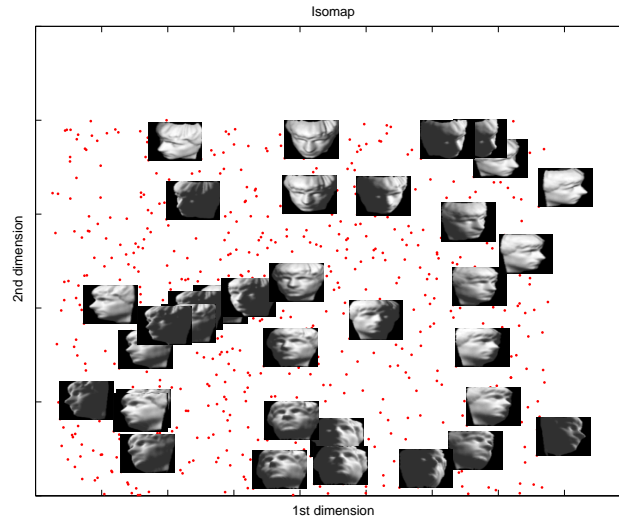


Figure 1.7: *Isomap applied ($k = 6$) to the same data set. A two-dimensional projection is shown, with a sample of the original input images.*

pairwise distances.

Like LLE, the Isomap algorithm proceeds in three steps:

1. Find the neighbours of each data point in high-dimensional data space.
2. Compute the geodesic pairwise distances between all points.
3. Embed the data via MDS so as to preserve these distances.

Again like LLE, the first step can be performed by identifying the k nearest neighbours, or by choosing all points within some fixed radius, ϵ . These neighbourhood relations are

represented by a graph G in which each data point is connected to its nearest neighbours, with edges of weight $d_X(i, j)$ between neighbours.

The geodesic distances $d_M(i, j)$ between all pairs of points on the manifold M are then estimated in the second step. Isomap approximates $d_M(i, j)$ as the shortest path distance $d_G(i, j)$ in the graph G . This can be done in different ways including Dijkstra's algorithm [29] and Floyd's algorithm [19].

These algorithms find matrix of graph distances $D^{(G)}$ contains the shortest path distance between all pairs of points in G . In its final step, Isomap applies classical MDS to $D^{(G)}$ to generate an embedding of the data in a d -dimensional Euclidean space Y . The global minimum of the cost function is obtained by setting the coordinates of y_i to the top d eigenvectors of the inner-product matrix B obtained from $D^{(G)}$

1.7 Semidefinite Embedding (SDE)

In 2004, Weinberger and Saul introduced semidefinite embedding (SDE) [35, 34] (See Figure 1.8 for an example). SDE can be seen as a variation on kernel PCA, in which the kernel matrix is also learned from the data. This is in contrast with classical kernel PCA which chooses a kernel function a priori. To derive SDE, Weinberger and Saul formulated the problem of learning the kernel matrix as an instance of semidefinite programming. Since

the kernel matrix K represents inner products of vectors in a Hilbert space it must be positive semidefinite. Also the kernel should be centered, *i.e.*, $\sum_{ij} K_{ij} = 0$. Finally, SDE imposes constraints on the kernel matrix to ensure that the distances and angles between points and their neighbours are preserved under the neighbourhood graph η . That is, if both x_i and x_j are neighbours (*i.e.*, $\eta_{ij} = 1$) or are common neighbours of another input (*i.e.*, $[\eta^T \eta]_{ij} > 0$), then the distance should be preserved

$$\|\Phi(x_i) - \Phi(x_j)\|^2 = \|x_i - x_j\|^2.$$

In terms of the kernel matrix, this constraint can be written as:

$$K_{ij} - 2K_{ij} + K_{jj} = \|x_i - x_j\|^2.$$

By adding an objective function to maximize $\text{Tr}(K)$ which represents the variance of the data points in the learned feature space, SDE constructs a semidefinite program for learning the kernel matrix K . The last detail of SDE is the construction of the neighbourhood graph η_{ij} . This graph is constructed by connecting the k nearest neighbours using a similarity function over the data, $\|x_i - x_j\|$. In its last step, SDE runs kernel PCA on learned kernel K . The algorithm is summarized in Algorithm SDE (Table 1.3).

Algorithm: SDE

Construct neighbours, η , using k -nearest neighbours.

Maximize $\text{Tr}(K)$ subject to $K \succeq 0$, $\sum_{ij} K_{ij} = 0$, and

$$\forall ij \quad \eta_{ij} > 0 \vee [\eta^T \eta]_{ij} > 0 \Rightarrow$$

$$K_{ii} - 2K_{ij} + K_{jj} = ||x_i - x_j||^2$$

Run Kernel PCA with learned kernel, K .

Table 1.3: *SDE Algorithm*

1.8 Unified Framework

All of the algorithms presented above can be cast as kernel PCA, which I now show.

Although this is obvious in some cases, it is less obvious for MDS, Isomap, LLE and Laplacian eigenmaps.

A straightforward connection between LLE and Kernel PCA has been shown in [26] and [39]. Let λ_{max} be the largest eigenvalue of $L = (I - W)^T(I - W)$. Then define the *LLE* kernel to be:

$$K_{LLE} = \lambda_{max}I - L \tag{1.5}$$

This kernel is, in fact, a similarity measure based on the similarity of the weights required to reconstruct two patterns in terms of k neighbouring patterns. The leading eigenvector of K_{LLE} is e , and the eigenvectors $2, \dots, d + 1$ provide the LLE embedding.

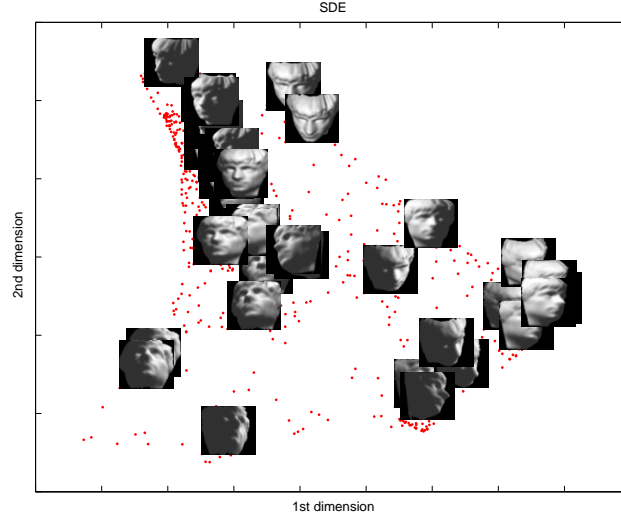


Figure 1.8: *SDE applied ($k = 5$) to the same data set. A two-dimensional projection is shown, with a sample of the original input images.*

An alternative interpretation of LLE as a specific form of Kernel PCA has been discussed in [15] in details. Based on this discussion, performing Kernel PCA on pseudo-inverse L^\dagger is equivalent to LLE up to scaling factors.

$$K_{LLE} = L^\dagger \quad (1.6)$$

Similarly Laplacian eigenmaps can be cast as Kernel PCA [15] by defining K_{LEM} as:

$$K_{LEM} = L^\dagger \quad (1.7)$$

where $L = R - W$, R is diagonal, and $R_{ii} = \sum_{j=1}^t W_{ij}$, as discussed in Section 1.4. K_{LEM} here is related to commute times of diffusion on the underlying graph.

It has been also shown [37] that metric MDS can be interpreted as kernel PCA. Given a distance matrix D , one can define K_{MDS} as:

$$K_{MDS} = -\frac{1}{2}(I - ee^T)D(I - ee^T) \quad (1.8)$$

where e is a column vector of all ones.

In the same fashion, given the geodesic distance $D^{(g)}$ used in Isomap, K_{Isomap} can be defined as [15]:

$$K_{Isomap} = -\frac{1}{2}(I - ee^T)D^{(g)}(I - ee^T) \quad (1.9)$$

The eigenvectors of (1.8) and (1.9) yield solutions identical to MDS and Isomap, up to scaling factor $\sqrt{\lambda_p}$, where λ_p is the p -th eigenvector.

The connection between kernel PCA and SDE is even more obvious. In fact, SDE is an instance of kernel PCA and the only difference is that SDE learns a kernel from data which is suitable for manifold discovery, while classical kernel PCA chose a kernel function a priori.

1.9 Thesis Contributions

One of the limitations of most contemporary dimensionality reduction techniques is that they optimize a single, generic criterion (i.e., minimizing discrepancy in MDS or maximizing

variance in PCA) that might not be suitable to capture the kind of variability one wishes to capture. Some mode of variation might be too subtle and therefore inconspicuous to a certain optimization criterion. As with any unsupervised learning method, embedding is only successful insofar as it captures some property of interest. With no information beyond the data itself, a given technique may uncover one or more properties but miss those that are of direct interest. This problem can be mitigated by properly exploiting *side information* for a given data set. For example, if labels are available for a subset of the data that reflect a characteristic of interest, the algorithm can be encouraged to exhibit that characteristic.

Recently many different approaches have been suggested for incorporating side information into unsupervised learning methods in order to influence their solutions. [8] proposes a method in order to preserve a user-defined grouping structure in PCA; [38] suggests pre-processing the input data to inform a model, where, a new distance metric is learned by considering limited information about the relationships between the points. [33] extracts two different kind of factors, by using bilinear models. All of these techniques require some grouping of the input data. That is, the side information is in the form of equivalence relations denoting data points that belong together.

In this thesis, I introduce three novel approaches to exploit and incorporate side information in the process of manifold discovery.

In Chapter 2, I present a dimensionality reduction technique that respects two different similarity measures simultaneously. This can be seen as a hybrid method which combines two different embedding techniques (e.g., combining LLE and Isomap, or Laplacian eigenmaps and MDS, etc.) with a *closed form solution* for the combined cost function. In this chapter, I cast the side information as similarity measurements and show how the extra information can be incorporated directly into the cost function of the embedding technique. The techniques I augment are mainly attractive because of the efficient computation afforded by the closed form solution for their cost optimization. This approach likewise produces a closed form solution for the augmented cost function. Empirical experiments show how we can incorporate valuable side information derived from transformation invariants on images. I present two different ways of incorporating transformation invariants in order to make new similarity measures.

Chapter 3 takes a more direct approach. It provides the embedding technique with some “hint” regarding a property of interest in the form of side information—limited information about the relationships between points. This chapter shows how two kinds of such side information can be used in a preprocessing step for embedding techniques, leading to embeddings that capture the target properties. The first kind of side information conveys information regarding the classes to which the data belongs, identifying pairs of points that belong to the same class or pairs that belong to different classes. The second kind of

side information takes the form of partial information about the similarity of points in the form of distances. It may be that available side information in a domain extends beyond the simple, boolean class-equivalence relation. One may have actual distances available for some pairs of points, obtained by some expensive measurement or derived as special cases from the nature of the domain. Chapter 3 shows how these two kinds of side information can be used to learn a new distance metric. The distances between points in this new space can then be used with any embedding technique.

In Chapter 4, I consider another form of side information present in domains where the data comes as the result of a sequence of discrete actions. A large number of problems of scientific interest, including the control of robots, industrial processes and inventory management, are sequential decision problems. In these problems, it is often the case that observations are given in a sequence along with actions that relate adjacent pairs of data points. Therefore, additional information is known about the data, i.e., the sequence of the observations and the actions between data points, but this information cannot be easily represented in the form of similarities. In this chapter, I present an algorithm that exploits this additional knowledge and greatly enhances manifold discovery. It is a variation on dimensionality reduction, where the output is a representation of the input data that is both low-dimensional and respects the actions (i.e., actions correspond to simple transformations in the output representation). The idea is to estimate a kernel matrix

that implicitly maps the original data space into a feature space, automatically discovering a mapping that unfolds the underlying manifold such that the actions correspond to simple transformations of points on the learned manifold. The computational method involves optimizing a positive semidefinite matrix by maximizing the variance in the feature space, subject to constraints that preserve the distances between nearest neighbours and consistency constraints derived from the action labels that relate observations.

Chapter 2

Hybrid Embedding

2.1 Introduction

In this chapter I develop a technique for combining different embedding objectives and then exploit this as a way to incorporate side information expressed in terms of transformation invariants that are known to hold in the data¹.

Recently Memisevic and Hinton [20] described a way of using multiple different types of similarity relationships to learn a low-dimensional embedding of a data set. In contrast to previous techniques, they allow the side information to be encoded in the form of similarity relations. In this chapter, as in [20], I cast the side information as similarity

¹Preliminary results of the work reported in this chapter have appeared in [14] and [13].

relations. However, the approach and optimization criteria differ substantially from those used in [20]. Here, I present a technique that allows the user to provide and combine more information than just a single set of similarities between data points. Although I use this technique as a way to incorporate transformation invariants as side information for image embedding, this technique can be seen as a general purpose framework for combining one locality-preserving method (LLE, Laplacian eigenmaps) and one distance-preserving technique (MDS, Isomap).

The proposed method augments existing embedding techniques with additional information provided by the user or derived from data. The extra information is incorporated directly into the cost function of the embedding technique. The techniques I augment are largely attractive because there is a *closed form solution* for their cost optimization. This approach likewise produces a closed form solution for the augmented cost function.

The remainder of this chapter is organized as follows. Section 2.2 derives a new embedding cost function to combine different sources of distance information and show a closed form solution for computing the embedding. To illustrate the combination technique, the effect of combining Isomap and LLE on a benchmark data set is then shown in Section 2.3. Section 2.4 then shows how this technique can be used to conveniently incorporate side information in the form of transformation invariants that are known to hold over the data a priori. Two different ways of incorporating transformation invariants to form a new

similarity measure are discussed in Section 2.5 and Section 2.6 respectively. I present the results of hybrid embedding, when these two new similarity measures are used as side information in Section 2.7, and finally conclude in Section 2.8.

2.2 Combined Embedding

I first present a new cost function that can combine two different similarity measures into a single cost function and show how a closed form solution can be obtained. One of the cost functions considers a distance-preserving embedding technique like MDS, and the other considers a locality-preserving method, like LLE or LEM.

We have seen before (Section 1.5) that Multidimensional scaling (MDS) finds a mapping from a high-dimensional space X to a lower-dimensional space Y while preserving pairwise distances between points. To achieve this, MDS must first convert a $t \times t$ distance matrix D to inner products, $M = X^T X = -\frac{1}{2} H D H^T$, where $H = I - \frac{1}{t} e e^T$ and e is a column vector of all 1's, and then optimize the following objective:

$$\min_Y \text{Tr}(M - Y^T Y)^2 \quad (2.1)$$

Recall that Isomap has the same objective as MDS (2.1). They only differ in the type of distance that they use. That is, in MDS D is the Euclidean distances between high-dimensional data points, while in Isomap, D is the geodesic distance. In our case D could

be any similarity relation different from Euclidean distances.

The second similarity measure I consider is expressed in a matrix, L , derived from a locality-preserving embedding method such as LLE or LEM. In this case L either equals $(I - W)^T(I - W)$, where W is the matrix of reconstruction weights (if one uses LLE), or L is the Laplacian (if one uses LEM). Both of these methods attempts to minimize local reconstruction error, and reduce to an objective of the form:

$$\min_Y \text{Tr}(Y^T Y L) \quad (2.2)$$

Combining (2.1) and (2.2) in a convex combination yields a minimization objective that considers both types of criteria simultaneously:

$$\min_Y (1 - \alpha) \text{Tr}(M - Y^T Y)^2 + \alpha \text{Tr}(Y^T Y L) \quad (2.3)$$

where Y is a matrix of code vectors and $0 \leq \alpha < 1$.² The first term in this objective is essentially the MDS objective, which tries to preserve the distances between data points (recall that the side information is encoded in a distance matrix D which is then transformed to a similarity kernel matrix M). The second term is the cost function of a locality-preserving embedding method. The parameter α mixes between the objectives, embedding solely on

²In practice, the matrix M should be rescaled to have a norm similar to L . The objective is to give the two parts of the objective function roughly equal scale so that α is more meaningful. This rescaling is omitted from the presentation for the sake of simplicity.

the basis of the distances as α tends to zero, or solely on the locality-preserving objective as α tends to 1.

Solving for the embedding Y in (2.3) can actually be done in closed form. In fact, this is the main advantage of the technique I propose in this chapter: the solution to the combined objective can be computed as efficiently as the solution to each of the component objectives separately. To solve this problem, I start by applying singular value decomposition (SVD) to obtain $M = V P V^T$, where V and P are the eigenvectors/values of M . If we also decompose $Y^T Y = Q \Lambda Q^T$ (Q and Λ are eigenvectors/values), then $Y = \Lambda^{\frac{1}{2}} Q^T$, allowing us to rewrite (2.3) as

$$\begin{aligned} & \min_{Q, \Lambda} (1 - \alpha) \text{Tr}(V P V^T - Q \Lambda Q^T)^2 + \alpha \text{Tr}(Q \Lambda Q^T L) \\ = & \min_{Q, \Lambda} (1 - \alpha) \text{Tr}(P - V^T Q \Lambda Q^T V)^2 + \alpha \text{Tr}(Q \Lambda Q^T L) \end{aligned}$$

If we now define $G = V^T Q$, we can replace $Q = V G$ to get

$$\begin{aligned} & \min_{G, \Lambda} (1 - \alpha) \text{Tr}(P - G \Lambda G^T)^2 + \alpha \text{Tr}(V G \Lambda G^T V^T L) \\ = & \min_{G, \Lambda} (1 - \alpha) \text{Tr}(P - G \Lambda G^T)^2 + \alpha \text{Tr}(V^T L V G \Lambda G^T) \\ = & \min_{G, \Lambda} \text{Tr}((1 - \alpha) P^2) + \text{Tr}((1 - \alpha) G \Lambda G^T G \Lambda G^T) \\ & - 2 \text{Tr}((1 - \alpha) P G \Lambda G^T) + \text{Tr}(\alpha V^T L V G \Lambda G^T) \end{aligned}$$

Note that P is a constant, so we can drop the first trace term when we are minimizing.

Collecting the third and fourth trace terms together, we get

$$\min_{G, \Lambda} \text{Tr}((1 - \alpha)G\Lambda G^T G\Lambda G^T) - 2\text{Tr}(BG\Lambda G^T)$$

where $B = (1 - \alpha)P - \frac{1}{2}\alpha V^T L V$. Note that B is constant, so we can add $\text{Tr}(\frac{1}{1-\alpha}B^2)$ to complete the square without affecting the minimization, obtaining (2.4), and then we can factor to produce the final objective (2.5)

$$\begin{aligned} \min_{G, \Lambda} \text{Tr}((1 - \alpha)G\Lambda G^T G\Lambda G^T) - 2\text{Tr}(BG\Lambda G^T) \\ + \text{Tr}(\frac{1}{1-\alpha}B^2) \end{aligned} \tag{2.4}$$

$$\begin{aligned} = & \min_{G, \Lambda} \text{Tr}(\frac{1}{\sqrt{1-\alpha}}B - \sqrt{1-\alpha}G\Lambda G^T)^2 \\ = & \frac{1}{\sqrt{1-\alpha}} \min_{G, \Lambda} \text{Tr}(B - (1 - \alpha)G\Lambda G^T)^2 \end{aligned} \tag{2.5}$$

This now has the same form as the standard MDS problem, so we can solve it by finding the decomposition $B = USU^T$. For a d -dimensional embedding, we set Λ to be the top d eigenvalues of S rescaled by $(1 - \alpha)$ and G to be the corresponding eigenvectors from U .³ We have now obtained a closed form solution to our mixed objective function.

While it will not be explored further here, it is worth noting that this derivation represents a general-purpose way to combine two embedding techniques into one. One of the techniques must have a distance-preserving-like cost function (e.g., MDS, Isomap), while

³When B is not positive semidefinite, one can add λI to B , where λ is the absolute value of the largest negative eigenvalue of S , or simply drop the negative eigenvalues in S .

the other must have a locality-preserving cost function (e.g., LLE, LEM). This means that one can potentially create a wide variety of hybrids that combine different similarity measures. Next, I will show examples of combining Isomap with LLE, before focusing on using transformation-derived invariant based side information to obtain improved embedding for image data.

2.3 Illustration: Combining Isomap and LLE

I begin with an easily visualized example. In this example, 2000 three-dimensional data points have been sampled from a “Swiss roll” [32]. Figure 2.1 shows these samples and the unfolded Swiss rolls learned by Isomap ($k = 12$) and LLE ($k = 12$). Figure 2.2 shows the effect of combining Isomap and LLE with different values of α . The parameter α mixes between Isomap and LLE, embedding like Isomap as α tends to zero, or like LLE as α tends to 1.

The Isomap-like embeddings tend to be “flatter” and more like a rectangle, while the more LLE-like embeddings concentrate points that lie further to the right along the x-axis. Notice how the shape smoothly interpolates between the two extremes as α changes.

Note that this example is only intended to illustrate the method. Combining locality-preserving techniques and distance-preserving techniques, where both use Euclidean dis-

tance as a similarity relation, may not be useful in practice. However if one of the techniques uses Euclidean distance as a similarity relation and the other use a different similarity relation, such as that derived from data or provided by user, then the combination can lead to improved embeddings that reflect more features of the data.

I will now exploit this technique to incorporate side information, by modifying the MDS based part of the objective.

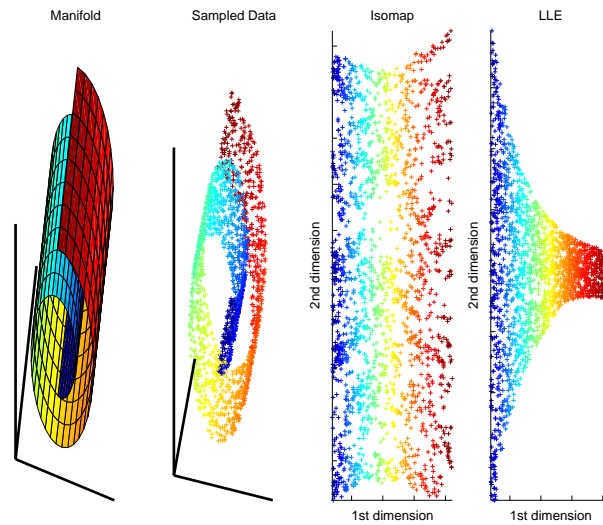


Figure 2.1: *From left to right: Actual manifold, 2000 data points sampled from the manifold, two-dimensional manifold found by Isomap, two-dimensional manifold found by LLE.*

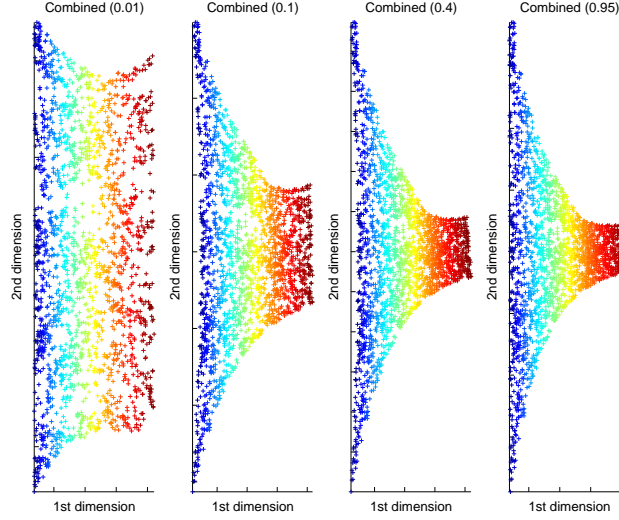


Figure 2.2: *Hybrids of Isomap and LLE with different α values. The value of α is shown in parentheses.*

2.4 Incorporating Transformation Invariants as Side Information

In the rest of this chapter, I will present an application of the introduced hybrid embedding technique on image data. Using this technique, we can now conveniently incorporate valuable side information encoded as transformation invariants on images.

Image data often has an underlying invariant and associated transformations, like rotation, that naturally imply a manifold on which neighbouring points are small transformations of one another. We can often characterize these transformations based on prior

knowledge regarding the source of the images (e.g., video data is likely to contain shifts, rotations, changes of illumination, etc). I introduce two new methods for exploiting the extra information offered by the transformations to correct potentially misleading observations based on Euclidean distance. The first considers angles between tangent spaces induced by the transformation. The second uses sequences of transformations that greedily minimize reconstruction error.

Other research has sought to use prior information regarding transformations, particularly with image data, in the context of clustering using probabilistic models that directly incorporate transformations [12], augmented distance measures in a supervised learning context [27], and for tracking [3]. Drawing from this body of research, I extend nonlinear embedding techniques to take advantage of known transformations, allowing for unsupervised learning of embeddings that better reflect the underlying dynamics.

2.4.1 Transformations

In many cases, we frequently have some prior knowledge regarding the transformations that relate images (e.g., rotation, translation, changes of illumination etc). In order to model the common transformations of image data, I consider three transformation functions here. The first, \mathcal{T}_{shift} , is a simple *shift* operator that translates the image by a fixed number of pixels in one direction (one pixel, horizontally, in experiments presented in this chapter).

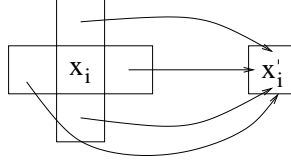


Figure 2.3: *Illustration of local pixel transformation to produce x' from x .*

This shift operator wraps around at the boundaries. The second transformation, \mathcal{T}_{rot} , is a fixed angle rotation about the center of the image. The third transformation, \mathcal{T}_{local} , is a more powerful transformation that can capture, translation, rotation and changes of illumination. This data-dependent transformation attempts to characterize the relationship between an image and its nearest Euclidean distance neighbour as a parameterized local filter applied over the whole image.

Let x be the original image and x' be its nearest Euclidean distance neighbour. Each pixel x'_i in x' is determined by a weighted combination of corresponding neighbouring pixels in the original image, $x'_i = \theta^T x_{N(i)}$, where $x_{N(i)}$ is a vector containing the pixels neighbouring x_i and θ is a weight vector. An illustration is given in Figure 2.3. When using this transformation, a weight vector is computed for each image that minimizes the Euclidean distance between the transformed original and the nearest neighbour to the original, $\min_{\theta} \|x' - \mathcal{T}_{local}(x, \theta)\|$. This is why I call it a “data-dependent” transformation.

2.5 Using Tangent Correction Distance

I now use transformational invariance to define a better way to measure distances between images for the purposes of embedding.

Consider a high-dimensional data set that lives on a lower-dimensional manifold. If the data lie densely along the manifold (e.g., very small steps of rotation) the Euclidean distances between consecutive points may be small enough to capture the manifold (see Figure 2.4 (a)). If, however, the data are sparsely distributed (Figure 2.4 (b)), the distances may be too large to be informative. Intuitively, one would wish to fill in the gaps along the manifold by generating new points likely to lie on the manifold. It is here that our known transformations (e.g., those described in Section 2.4.1 or some other transformation implied by the domain) come into play.

One strategy is to generate “virtual” points by applying transformations to real data points and then learning the manifold using both the real and the virtual points (Figure 2.4 (c)). Here, however, rather than explicitly creating points and adding them, I approximate the *tangent space* at the real points, and use the angle between the tangent spaces of two points as a measure of similarity between those points. The intuition behind this treatment of these tangent spaces is as follows. If the tangents of two points x_i and x_j are unaligned (as in Figure 2.5 (b)) rather than aligned (as in Figure 2.5 (a)), the Euclidean distance is probably inaccurate. While there is no guarantee that the Euclidean distance between

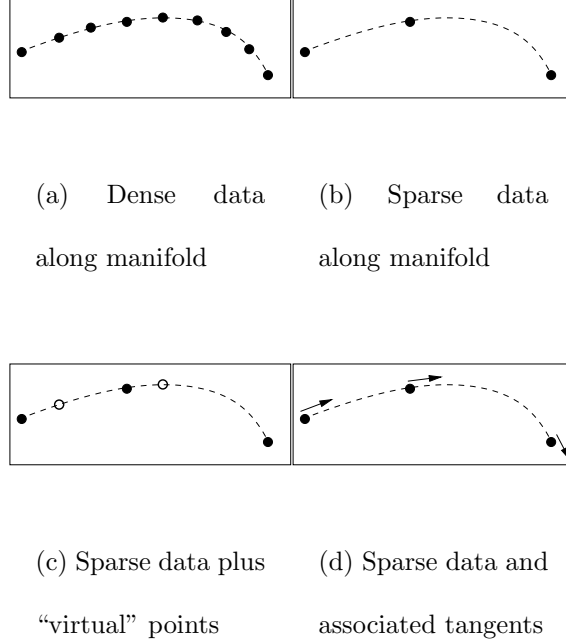


Figure 2.4: *Data and tangents along the manifold*

aligned points is accurate for points that are very distant along the manifold (there may be many curves along the way), over smaller distances the alignment provides some good evidence of the reliability of Euclidean distance. I therefore use the angle between tangent spaces as a correction to the Euclidean distance in the embedding cost functions. A simple way to do this is to compute the sine of the angle between each pair of tangent spaces and treat this as a new measure of distance. The sine function gives us a number between 0 and 1 for each pair, where 0 means the pair is parallel (aligned) and 1 means the pair is orthogonal (unaligned). More formally, I construct a matrix, $D^{(\mathcal{T})}$, containing the sine

squared of the angle between each pair of tangent spaces.

The tangent space of the manifold at point x_i can be approximated as follows. Suppose there are t points $x_1 \dots x_t$ in high-dimensional space that need to be embedded in low-dimensional space. A large group of manifold learning methods including LLE, Isomap, Laplacian eigenmaps and Semi-Definite Embedding do this task by building a $t \times t$ data-dependent kernel matrix, K , and computing its eigenvectors. While these techniques differ in the way that they produce their data-dependent kernels, they are similar in their final step, that is, sorting the eigenvectors of K by their eigenvalues and selecting the largest d eigenvectors as the embedding (typically $d \ll t$). More formally assume (v_k, λ_k) is an eigenvector/value pair of K . If we assemble the top d eigenvectors as the columns of a $t \times d$ matrix E , the d -dimensional code vector y_i corresponding to x_i is the i -th row of E . Given such an embedding, we can now consider the tangent space at a point on this low-dimensional manifold. As described in [2], the tangent space at x_i is the subspace spanned by the vectors $\frac{\partial v_k}{\partial x_i}$. It can be shown that this derivative depends only on the near neighbours of x_i . In fact, $\frac{\partial v_k}{\partial x_i}$ can be closely approximated by a linear combination of vectors $(x_i - x_k)$, where x_k is a neighbour of x_i . Thus, the span of the vectors $(x_i - x_k)$ form an approximation of the manifold tangent space at x_i .

When the data lie densely along the manifold, one can simply choose nearest neighbours of x_i from the training data and span an approximation of the tangent space by $(x_i -$

x_k) vectors. However, when the data are sparsely distributed, such an approximation might be very inaccurate. In this case, we can take advantage of our transformation. Consider a transformation, $\mathcal{T}(x, \theta)$, parameterized by θ (e.g., θ could be the angle in a rotation). We can, instead of choosing x_k 's from the training set, use the transformation to generate virtual points $\tilde{x}_k = \mathcal{T}(x_i, \theta)$ close to x_i (i.e., for small values of θ). The underlying assumption here is that the transformation locally characterizes the manifold, so that tangents to the transformation function approximate tangents to the manifold (Figure 2.4 (d)). The tangent information so obtained can be combined, as described earlier, with Euclidean distances to produce a better embedding.

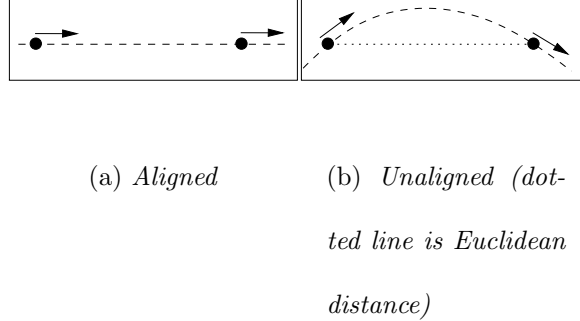


Figure 2.5: *Alignment of tangent vectors*

2.6 Transformation Reconstruction Distance

An alternative way to use transformations in image comparison is to apply a transformation to an image i and then measure the distance of this transformed image to another image j . I call this distance the *reconstruction error*, representing how well image i can be used to reconstruct the target image j using only a single application of some known transformation. If the target image j is truly a transformed version of the original, then the corresponding reconstruction error should be very small.

In general, if images on the manifold were the result of repeated applications of the transformation, it should be possible find a sequence of images in the data set with very small reconstruction error at each step in the sequence. Finding such a sequence can be seen as a shortest path problem on a graph weighted by the reconstruction errors. This is similar to the approach used by Isomap to obtain so-called geodesic distances [32]. I call the sum of the reconstruction errors along the path the *transformation reconstruction distance*. These distances can then be used to “correct” Euclidean distance as described in Section 2.2.

2.7 Results

Both forms of side information described here (tangent correction distance and transformation reconstruction distance) can be used as corrections to Euclidean distance via any embedding method that has a cost function similar in form to locally linear embedding (LLE) and the Laplacian eigenmaps method (LEM). In order to demonstrate the effectiveness of correcting using side information in this way, experimental results are presented with and without the corrections. These embeddings are examined for useful structure, and the presence of such structure in the corrected embeddings compared to the uncorrected embeddings shows that the method is effective.

I experimented with embeddings for a variety of images using the following methods: LLE, Isomap, LEM, and *transformation-corrected LEM* (TC-LEM) with the tangent correction and the transformation reconstruction correction, respectively. These last two methods correspond to using the combined embedding objective (2.3), where L is LEM's Laplacian matrix. Each method was tried with a variety of neighbourhood settings and the best chosen. In most cases LEM, LLE, and Isomap behaved very similarly so I use LEM here as a representative for comparison. Parameters are specified in parentheses after the method (i.e., $\text{LEM}(k)$ and $\text{TC-LEM}(k, \alpha)$, where k is the number of neighbours used to build the Laplacian). Note that $\text{TC-LEM}(k, 1)$ is effectively identical to $\text{LEM}(k)$.

Figure 2.6 shows the 2-D embedding using of a set of three handwritten digit images

(5, 6, and 8 - see Figure 2.10 for the raw images), with nine images of each digit, each image shifted horizontally against a white background. Plotting the actual images leads to overlaps, so the plot only shows the number associated with each digit and indicates the degree of shift by the darkness or lightness of the digit (darkest - leftmost and lightest - rightmost). At first glance the manifold produced by LEM seems informative, and it has indeed captured the shift of the images along the horizontal axis. However, the other dimension is essentially meaningless, and LEM is incapable of distinguishing the digits. By contrast, TC-LEM, using \mathcal{T}_{local} and tangent correction distance, captures the shift of the images on the vertical axis and also manages to separate the three digits effectively along the horizontal axis.

Figure 2.7 show similar results for all ten digits. LEM again captures the shift but fails to distinguish the digits. I show TC-LEM run with \mathcal{T}_{local} and two different α parameters for comparison. When $\alpha = 0.45$, TC-LEM tends to “cluster” the data, producing concentrations of points corresponding to each image. By blowing up a section of this plot (Figure 2.9 - left), we can see that the shift aspect of the data is still somewhat represented in the horizontal access. When $\alpha = 0.65$, TC-LEM spreads the different digits out across the horizontal axis while showing the shift along the vertical access. A blowup of the densest part of the plot (Figure 2.9 - right) shows that there is still decent separation between digits. Remembering that $\text{TC-LEM}(k,1)$ is the same as $\text{LEM}(k)$, we can see that for large

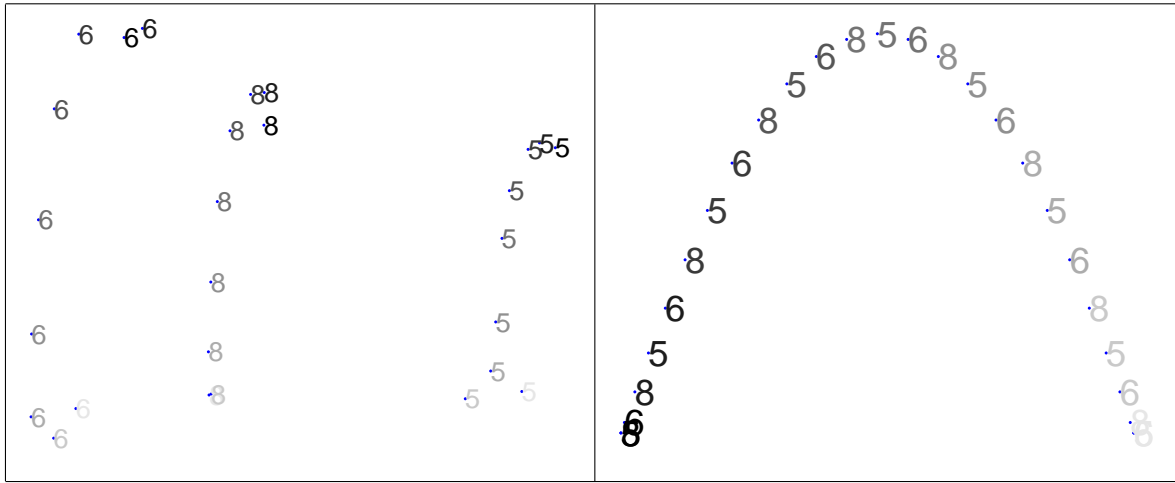


Figure 2.6: *Schematic plot of two-dimensional manifold found by (left) TC-LEM(5, 0.45) and (right) LEM(2) for images of three digits (5, 6, and 8) with multiple images for each digit shifted horizontally by varying amounts. Actual images are not plotted but the shift is indicated by the darkness of the digit (leftmost - darkest; rightmost - lightest).*

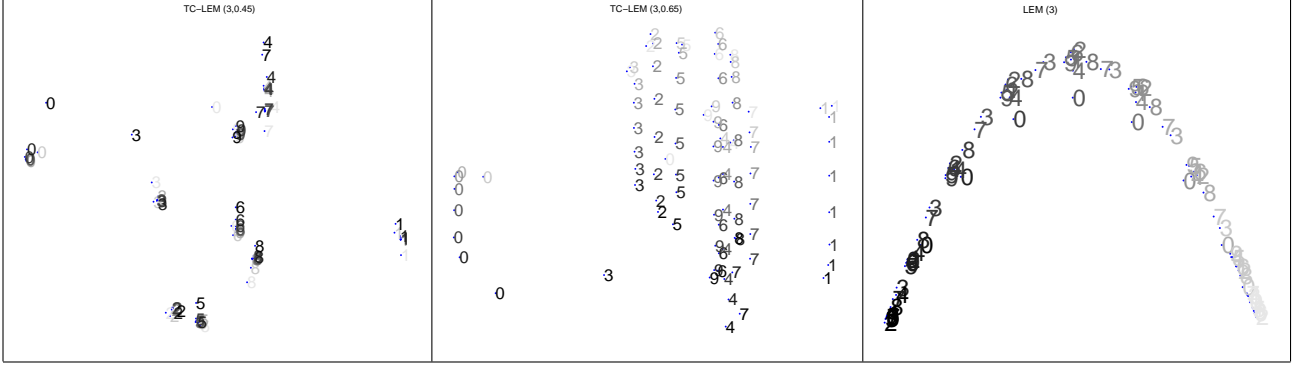


Figure 2.7: *Schematic plot of two-dimensional manifold found by (from left to right) $TC-LEM(3,0.45)$, $TC-LEM(3,0.65)$ and $LEM(3)$ for images of all ten digits with multiple images for each digit shifted horizontally by varying amounts. Actual images are not plotted but the shift is indicated by the darkness of the digit (leftmost - darkest; rightmost - lightest).*

alphas, the method loses any ability to cluster the digits, and can only capture the overall shift.

This tendency toward clustering vs. sequencing is consistent in all experience and suggests that TC-LEM may be viewed as simultaneously trying to cluster and embed. The α parameter can be used to control this tendency in a straightforward fashion.

Figure 2.12 shows similar results for TC-LEM using \mathcal{T}_{shift} and the reconstruction transformation distance. Again, TC-LEM is able to cluster the digits while still capturing the shift, although there is some overlap in the densest region.

Figure 2.13 show the 2-D embedding of a set of eighteen images of a teapot. The teapot

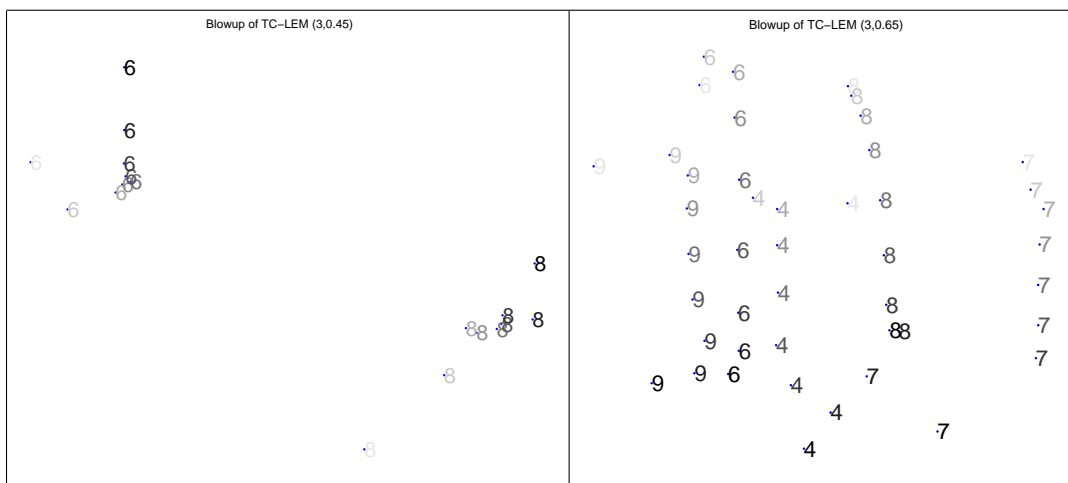


Figure 2.8: *Blowup of sections of TC-LEM results from Figure 2.7.*

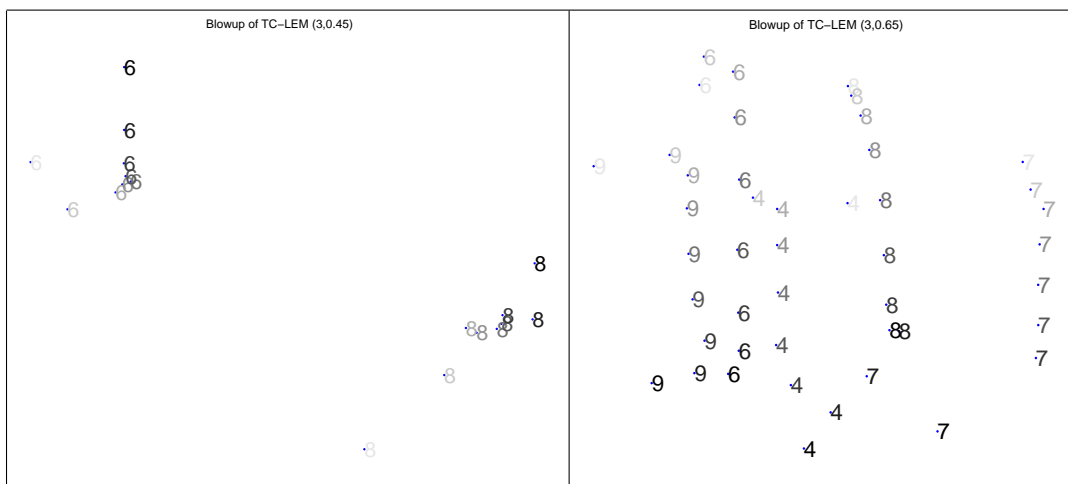


Figure 2.9: *Blowup of sections of TC-LEM results from Figure 2.7.*

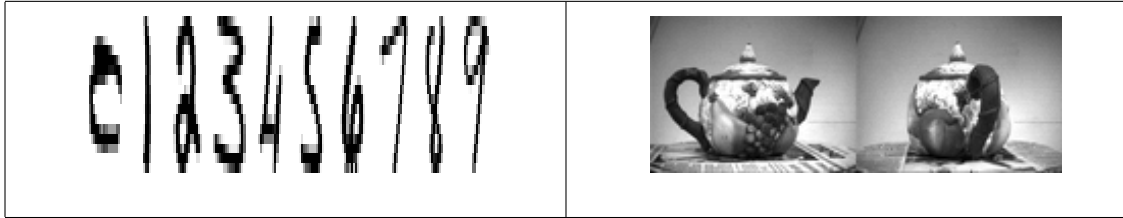


Figure 2.10: *Original digit and teapot images used in experiments.*

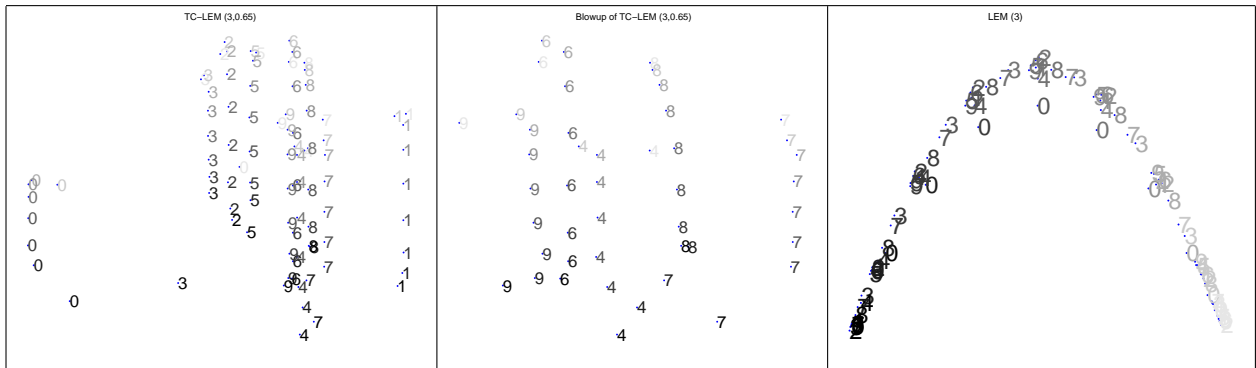


Figure 2.11: *Schematic plot of two-dimensional manifold found by (from left to right) TC-LEM(3, 0.65) using T_{local} and tangent correction (left), a blowup of the dense part of that plot (middle), and LEM(3) (right) for images of all ten digits with multiple images for each digit shifted horizontally by varying amounts. Actual images are not plotted but the shift is indicated by the darkness of the digit (leftmost - darkest; rightmost - lightest).*

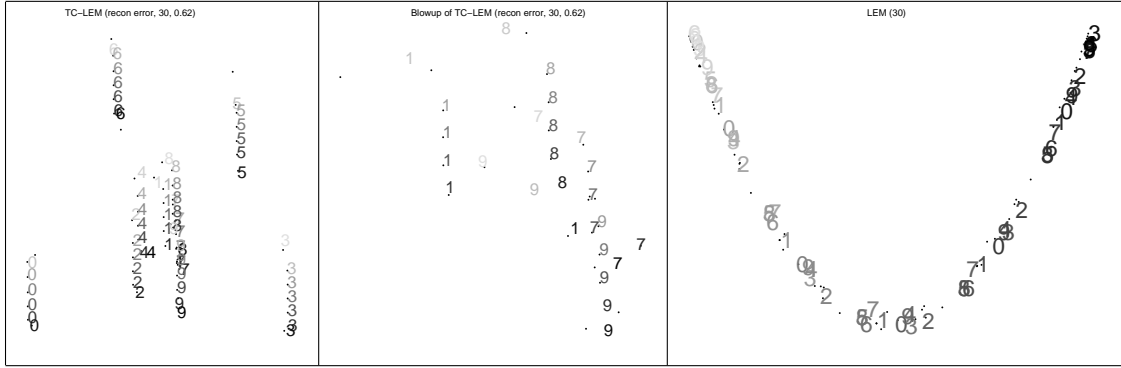


Figure 2.12: *Schematic plot of two-dimensional manifold found by (from left to right) TC-LEM(30, 0.62) with \mathcal{T}_{shift} and reconstruction error (left), a blowup of the dense part of that plot (middle), and LEM(30) (right) for images of all ten digits with multiple images for each digit shifted horizontally in one pixel increments. Actual images are not plotted but the shift is indicated by the darkness of the digit (leftmost - darkest; rightmost - lightest). For readability, only every fifth digit position is plotted.*

is viewed from two different angles (see Figure 2.10 for the raw images) and each subset is rotated in the plane in nine steps through to 180 degrees. LEM captures the rotation (along the horizontal axis) but fails to distinguish between the two views (the vertical axis has no readily apparent meaning and the two sets overlap almost perfectly). TC-LEM, using \mathcal{T}_{local} and tangent correction, captures both rotation in the plane (as a “loop” of images) and the two distinct views of the teapot (there are two loops). Figure 2.14 shows results for \mathcal{T}_{rot} and transformation reconstruction distance. In this case, however, it has not clustered as effectively, but has still separated the two teapots except at one orientation.

In general, all methods are capable of identifying meaningful relationships in the data but TC-LEM is reliably capable of capturing both the dimension corresponding to the sequencing of images and the other distinguishing features (e.g., different objects).

2.8 Conclusions

I have developed an algorithm that combines different types of similarity measures and still produces an embedding corresponding to the underlying structure of the data. I have demonstrated that known transformations inherent in the image domain can be used to augment nonlinear embedding techniques by correcting potentially misleading Euclidean distances. Transformation corrected embeddings reliably capture the dimension corre-

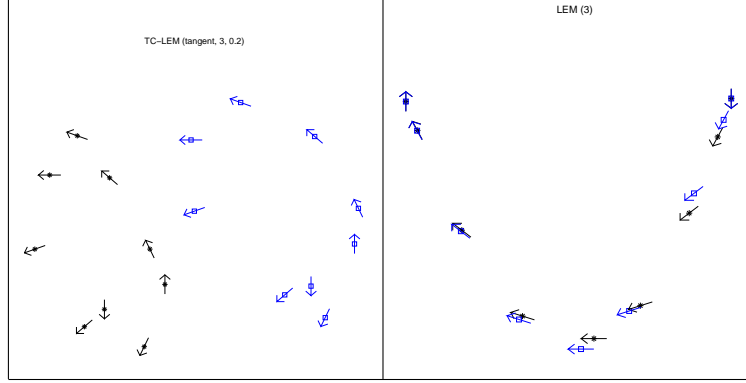


Figure 2.13: *Schematic plot of two-dimensional manifold found by (left) TC-LEM(0.2) using \mathcal{T}_{local} and tangent correction, and (right) LEM for images of a teapot viewed from two different angles and rotated in the plane. Arrows indicate the angle of rotation for each image and points (square or star) indicate the teapot image.*

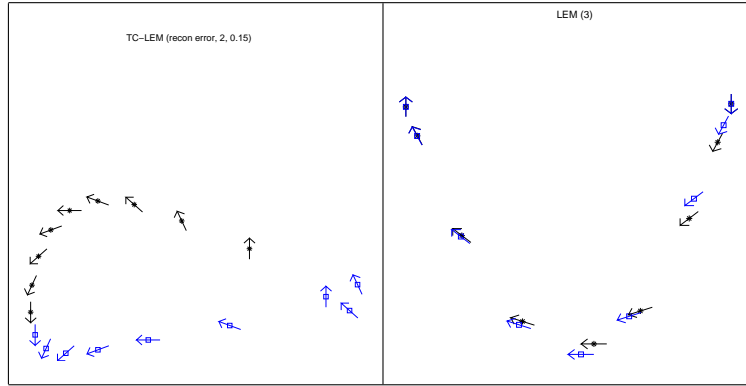


Figure 2.14: *Schematic plot of two-dimensional manifold found by (left) TC-LEM(0.15) using \mathcal{T}_{rot} and transformation reconstruction distance, and (right) LEM for images of a teapot viewed from two different angles and rotated in the plane.*

sponding to a sequence of such transformations and other distinguishing features along with it. Computationally, the method retains the advantages of the techniques we augment by having a closed form solution for the optimization and represents a general purpose method for combining distance-preserving and locality-preserving embedding methods. Future work consists in extending the approach to use multiple transformations simultaneously, more sophisticated methods for characterizing the local manifold, and trying new combinations of embedding techniques.

Chapter 3

Learning a Metric for Embedding

3.1 Introduction

In this chapter I consider different types of side information for learning a metric which can then be used in a subsequent embedding method.

It may be possible to obtain a small amount of information regarding the similarity of points in a particular data set. Consider a large collection of images. While it would be expensive to have a human examine and label the entire set, it would be practical to have a small subset and provide information on how they relate to each other. This chapter will show how two kinds of such side information can be used in a preprocessing step for embedding techniques, leading to embeddings that capture the target properties.

The first kind of side information relates to classes to which the data belongs, identifying pairs of points that belong to the same class or pairs that belong to different classes. Note that this information is about the class-equivalence/inequivalence of points but does not give the actual class labels. Consider a case where there are four points, x_1, x_2, x_3 , and x_4 . Given side information that x_1 and x_2 are in the same class, and x_3 and x_4 also share a class, we still cannot be certain whether the four points fall into one or two classes.

The second kind of side information takes the form of partial information about the similarity of points in the form of distances. It may be that available side information in a domain extends beyond the simple, boolean class-equivalence relation. One may have actual distances available for some pairs of points, obtained by some expensive measurement or derived as special cases from the nature of the domain. Molecular conformation problems are examples of such a domain. Some of the distances between pairs of atoms in a given molecule can be determined by nuclear magnetic resonance spectroscopy, but the procedure is costly and certainly not guaranteed to provide all such distances. Determining the rest of these distances can be invaluable in classifying the conformation of the molecule (see [10] for more details).

These two kinds of side information can be used to learn a new distance metric. The distances between points in this new space can be used with any embedding technique. I start by showing how class-equivalence side information can be used to learn such a

metric in Section 3.2 and 3.2.1 . The effect of using this new metric in various embedding techniques are presented in Section 3.2.2. Then I show how multiple applications of this method can be used to combine several learned distance metrics together into one in order to capture multiple attributes in the embedding in Section 3.3, follows by the experimental results of this technique in Section 3.3.1. Section 3.4 extends the approach of learning class-equivalence side information by kernelizing it, allowing for nonlinear transformations of the metric. Finally, I formulate a method for using the second type of side information, where we have partial information about desirable target distances between some pairs of points in Section 3.5. Experimental results demonstrate the value of this preprocessing step in Section 3.5.1 before the conclusion in Section 3.6.

3.2 Learning a Metric from Class-Equivalence Side Information

I will start with the simpler similar/dissimilar pair case, formalizing this notion of side information and stating an objective that will be optimized using standard semidefinite programming software.

One method along these lines is described by Xing *et al.* [38]. In this work, a new distance metric is learned by considering side information. Xing *et al.* used side information

identifying pairs of points as “similar”. They then construct a metric that minimizes the distance between all such pairs of points. At the same time, they attempt to ensure that all “dissimilar” points are separated by some minimal distance. By default, they consider all points not explicitly identified as similar to be dissimilar. They present algorithms for optimizing this objective and show results using the learned distance for clustering, an application in which an appropriate distance metric is crucial.

The use of this kind of side information allows one to select the characteristic for distinction. For example, one may have several images of men and women, with and without glasses. One sensible cluster is by gender. Another is by the presence or absence of glasses. Different indications of similarity allow the capturing of either distinction.

The work presented here takes the same basic approach but offers a simpler optimization procedure using “off-the-shelf” optimization methods instead of the iterative method described by Xing *et al.*

3.2.1 Derivation

Given a set of t points, $\{x_i\}_{i=1}^t \subseteq R^n$, I identify two kinds of class-related side information. The first is a set of pairs of similar or class-equivalent points (they belong to the same class)

$$S : (x_i, x_j) \in \mathcal{S} \text{ if } x_i \text{ and } x_j \text{ are similar}$$

and the second is a set of dissimilar or class-inequivalent pairs (they belong to different classes)

$$\mathcal{O} : (x_i, x_j) \in \mathcal{O} \text{ if } x_i \text{ and } x_j \text{ are dissimilar}$$

We then wish to learn a matrix A that induces a distance metric $D^{(A)}$ over the points

$$D^{(A)}(x_i, x_j) = \|x_i - x_j\|_A = \sqrt{(x_i - x_j)^T A (x_i - x_j)}$$

where $A \succeq 0$.

I define the following loss function, which, when minimized, attempts to minimize the squared induced distance between similar points and maximize the squared induced distance between dissimilar points

$$L(A) = \sum_{(x_i, x_j) \in \mathcal{S}} \|x_i - x_j\|_A^2 - \sum_{(x_i, x_j) \in \mathcal{O}} \|x_i - x_j\|_A^2$$

The optimization problem then becomes

$$\begin{aligned} & \min_A L(A) \\ \text{s.t. } & A \succeq 0 \\ & \text{Tr}(A) = 1 \end{aligned} \tag{3.1}$$

The first constraint (positive semidefiniteness) ensures a valid Euclidean metric. The second constraint excludes the trivial solution where all distances are zero. The constant in this constraint is arbitrary and changing it simply scales the resulting space.

This objective will be optimized using standard semidefinite programming software and so it must be converted to a linear objective. Expanding the loss function

$$L(A) = \sum_{(x_i, x_j) \in \mathcal{S}} (x_i - x_j)^T A (x_i - x_j) - \sum_{(x_i, x_j) \in \mathcal{O}} (x_i - x_j)^T A (x_i - x_j)$$

each squared distance term must be converted. I start by observing that $\text{vec}(XYZ) = (Z^T \otimes X)\text{vec}(Y)$, where $\text{vec}()$ simply rearranges a matrix into a vector by concatenating columns and \otimes is the Kronecker product. Note that $(x_i - x_j)^T A (x_i - x_j) = \text{vec}((x_i - x_j)^T A (x_i - x_j))$ because the left-hand side is a scalar. Using this and the fact that $(a^T \otimes b^T) = \text{vec}(ba^T)^T$, I can rewrite the squared distance terms as

$$\begin{aligned} (x_i - x_j)^T A (x_i - x_j) &= \text{vec}((x_i - x_j)^T A (x_i - x_j)) \\ &= ((x_i - x_j)^T \otimes (x_i - x_j)^T) \text{vec}(A) \\ &= \text{vec}((x_i - x_j)(x_i - x_j)^T)^T \text{vec}(A) \\ &= \text{vec}(A)^T \text{vec}((x_i - x_j)(x_i - x_j)^T) \end{aligned}$$

The linear loss function is then

$$\begin{aligned}
L(A) &= \sum_{(x_i, x_j) \in \mathcal{S}} \text{vec}(A)^T \text{vec}((x_i - x_j)(x_i - x_j)^T) - \sum_{(x_i, x_j) \in \mathcal{O}} \text{vec}(A)^T \text{vec}((x_i - x_j)(x_i - x_j)^T) \\
&= \text{vec}(A)^T \left[\sum_{(x_i, x_j) \in \mathcal{S}} \text{vec}((x_i - x_j)(x_i - x_j)^T) - \sum_{(x_i, x_j) \in \mathcal{O}} \text{vec}((x_i - x_j)(x_i - x_j)^T) \right]
\end{aligned}$$

This form, along with the two constraints from (3.1), can be readily submitted to an SDP solver to optimize the matrix A ¹. Aside from this convenient form, this formulation has other advantages over that used by Xing *et al.*, especially with respect to the side information I can convey. Xing *et al.* require at least one dissimilar pair in order to avoid the trivial solution where all distances are zero. The constraint on the trace that I employ means that do not place any restrictions on pairings. There can be only similar pairs, only dissimilar pairs, or any combination of the two, and the method will still avoid trivial solutions.

Furthermore, in the absence of specific information regarding dissimilarities, Xing *et al.* assume that all points not explicitly identified as similar are dissimilar. This information may be misleading, forcing the algorithm to separate points that should in fact be similar. The formulation presented here allows one to specify only the side information one actually has, partitioning the pairings into similar, dissimilar, and unknown.

¹I use the MATLAB SDP solver SeDuMi [28]

3.2.2 Results

Once a metric has been learned, one can use the new distances in any embedding technique by feeding it the transformed distances (or transformed points) instead of the raw data. The benefits of the approach must now be demonstrated. To do so, embeddings are generated with and without the preprocessing step, using side information to inform the preprocessed embeddings about some characteristic of interest. The embeddings can then be examined to see whether they capture that characteristic. To show that the side information truly informs the embedding, two sets of informed embeddings are generated, each with side information regarding two different characteristics. Structure in the resulting embeddings that captures the two different characteristics within a single data set is evidence that the method works as intended.

The results shown in Figures 3.1-3.10 and A.1-A.10 (in Appendix A) demonstrate the benefits of the preprocessor on image data for a variety of embedding techniques including MDS, Isomap, LEM, LLE, and SDE. There are 200 images of faces and two distinctions are identified by side information: faces with beards vs. faces without beards and faces with glasses vs. faces without glasses².

In one set of experiments, all similar pairs were identified but no dissimilar pairs. The

²For all methods in this chapter, the results for faces with glasses vs. faces without glasses are demonstrated in Appendix A

second set, simulating a situation where labelling is expensive, identifies only four similar pairs and one dissimilar pair. The pairs were selected at random. Techniques using the preprocessor are labelled as “Equivalence-Informed” (e.g., Equivalence-Informed MDS). In each case, a two-dimensional embedding is shown. The two clusters in each case are marked as an X or an O, as appropriate. Additionally, a subset of the images are displayed on the plot (including all images renders the plot unreadable).

Note that, in general, the informed versions of these embeddings manage to separate the data based on the target property, whereas the uninformed versions are typically chaotic. Even when separation is poor in the informed embedding, there is still typically much more structure than in the uninformed embedding. Equivalence-Informed MDS (Figures 3.1, A.1, 3.2 and A.2) offers mixed results, especially with limited side information for glasses vs. no glasses (Figure A.2), but MDS is a comparatively crude embedding technique in any case. Isomap with all similar pairs identified works very well (Figures 3.3 and A.3) and with limited side information, still manages to group the two classes, but does not separate them well. Figure 3.5 and A.5 show good separation for LEM with all similar pairs, and effective grouping with inferior separation in the limited information case (Figures 3.6 and A.6). The same effective grouping and varying separation occurs for LLE (Figures 3.7, A.7, 3.8, and A.8). Finally, SDE groups well, but is somewhat weak in separation in all cases (Figures 3.9, A.9, 3.10, and A.10).

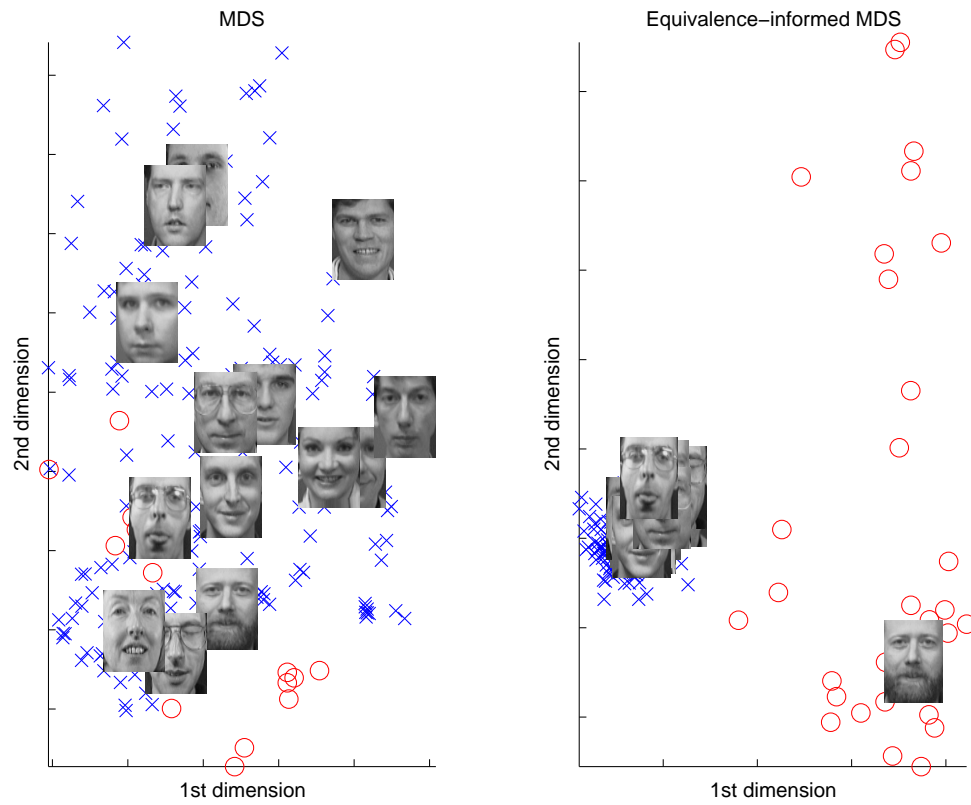


Figure 3.1: *MDS and Equivalence-Informed MDS with bearded/unbearded distinction (all class-equivalent pairs)*

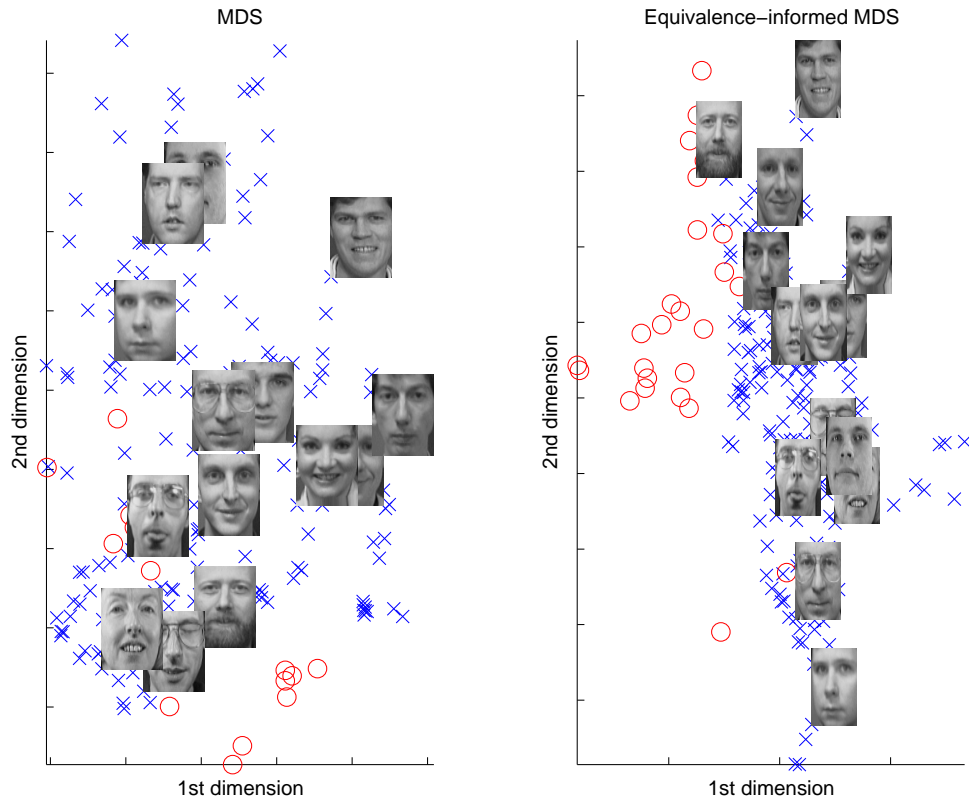


Figure 3.2: *MDS and Equivalence-Informed MDS with bearded/unbearded distinction (four class-equivalent pairs, one class-inequivalent pair)*

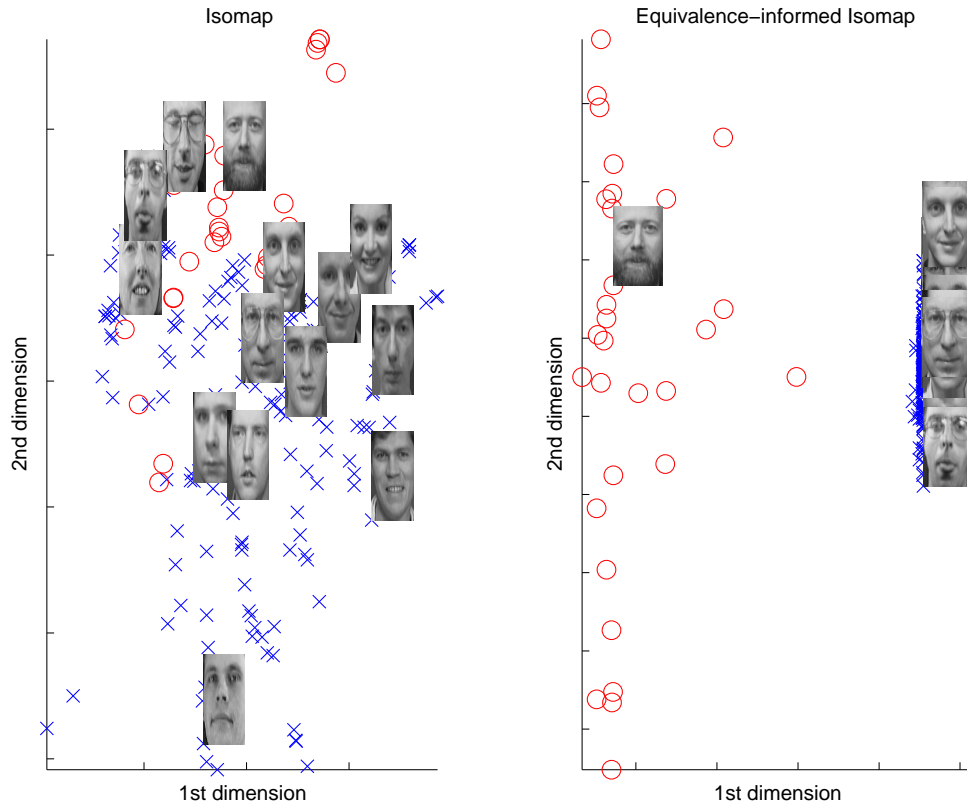


Figure 3.3: *Isomap and Equivalence-Informed Isomap with bearded/unbearded distinction (all class-equivalent pairs)*

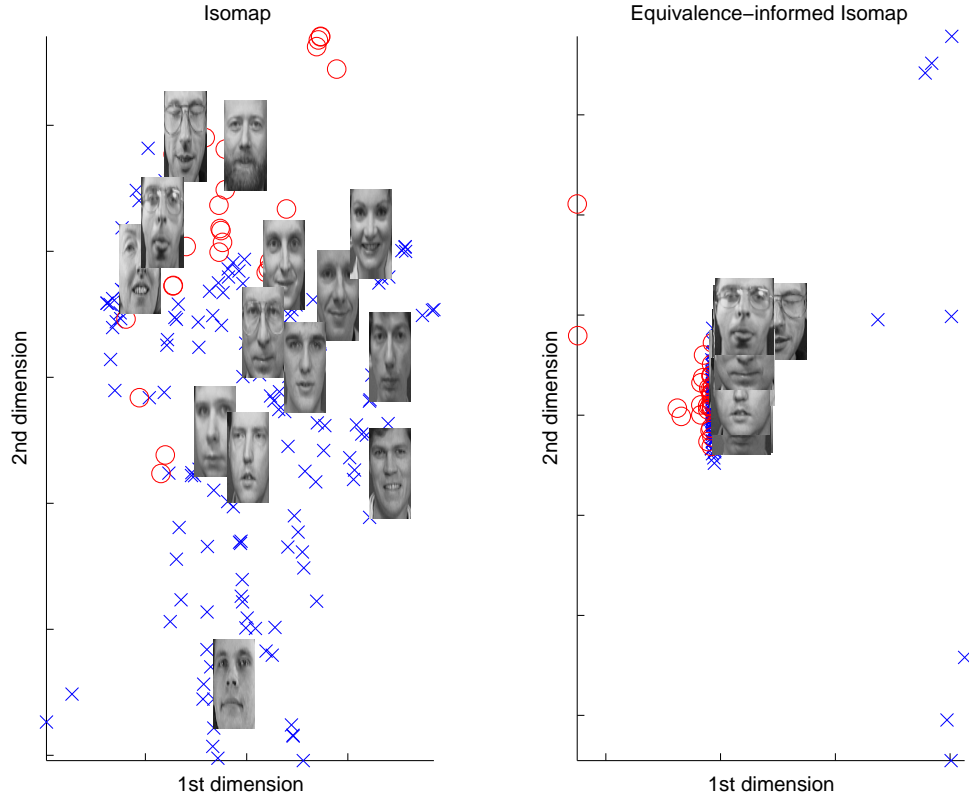


Figure 3.4: *Isomap and Equivalence-Informed Isomap with bearded/unbearded distinction*
(four class-equivalent pairs, one class-inequivalent pair)

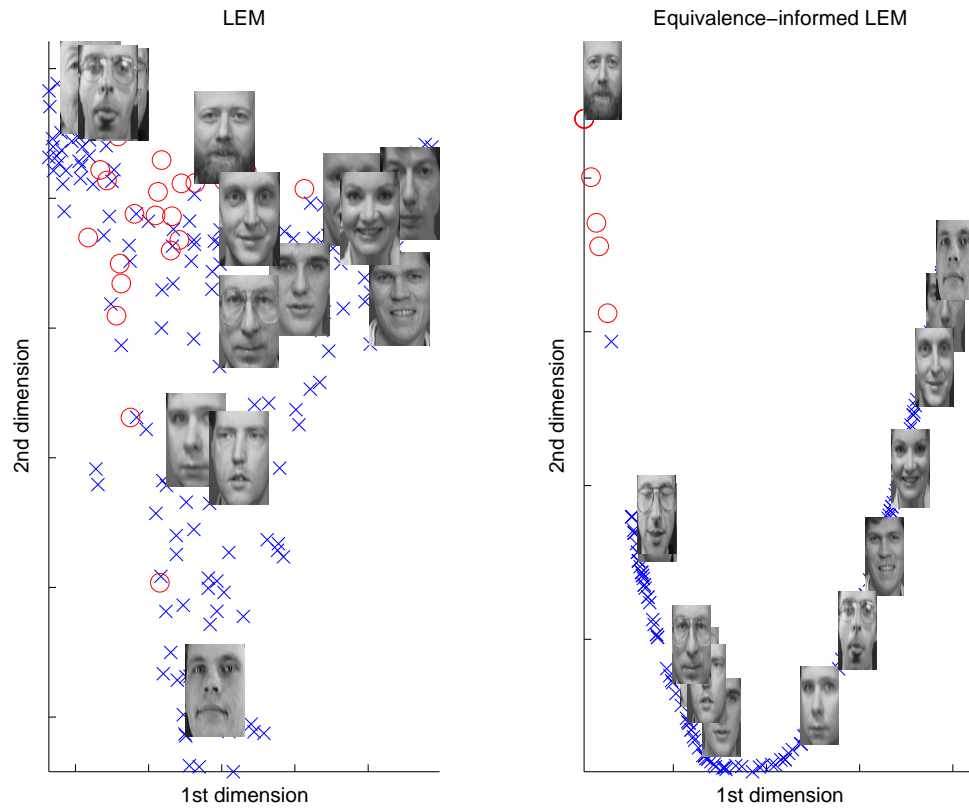


Figure 3.5: *LEM and Equivalence-Informed LEM with bearded/unbearded distinction (all class-equivalent pairs)*

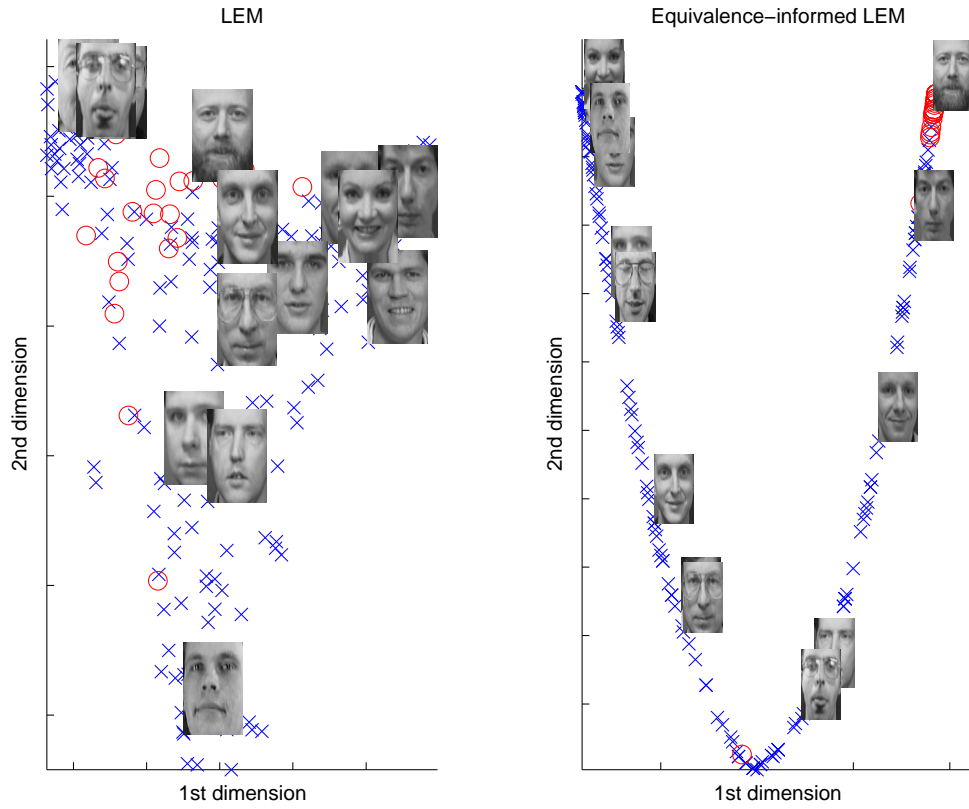


Figure 3.6: *LEM and Equivalence-Informed LEM with bearded/unbearded distinction (four class-equivalent pairs, one class-inequivalent pair)*

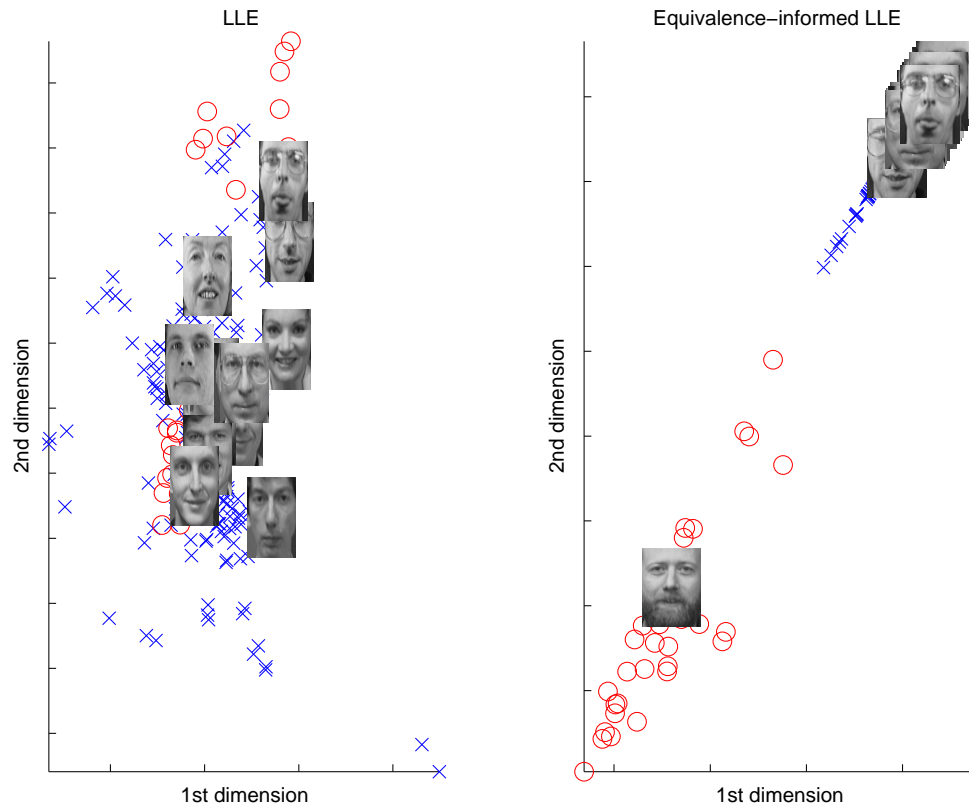


Figure 3.7: *LLE and Equivalence-Informed LLE with bearded/unbearded distinction (all class-equivalent pairs)*

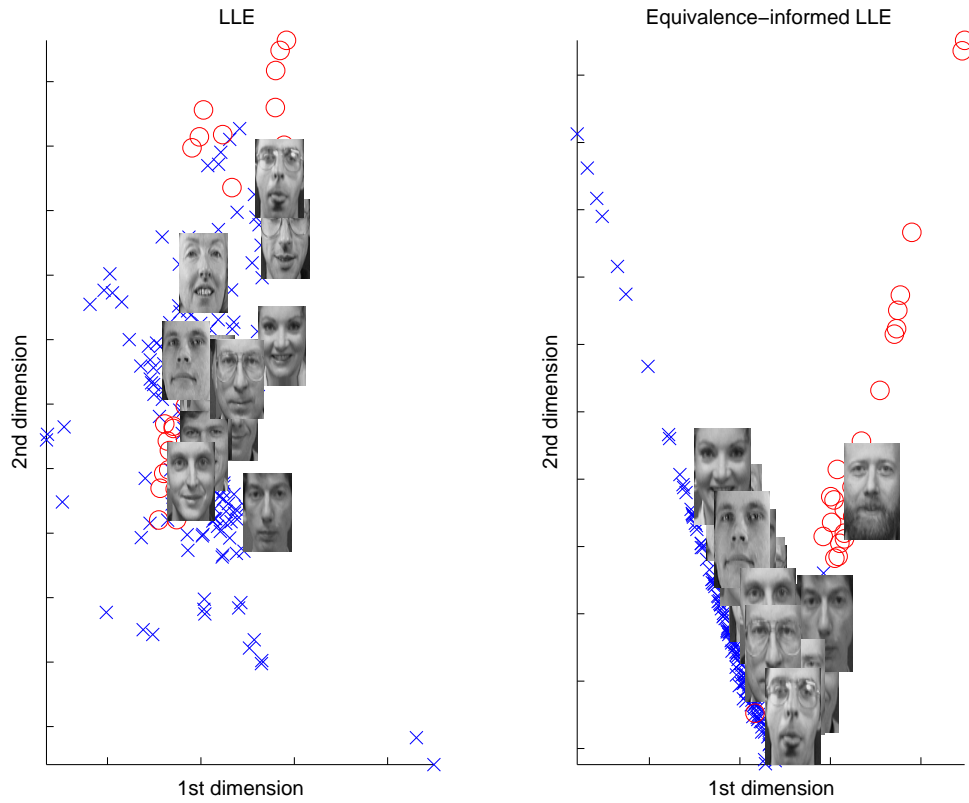


Figure 3.8: *LLE and Equivalence-Informed LLE with bearded/unbearded distinction (four class-equivalent pairs, one class-inequivalent pair)*

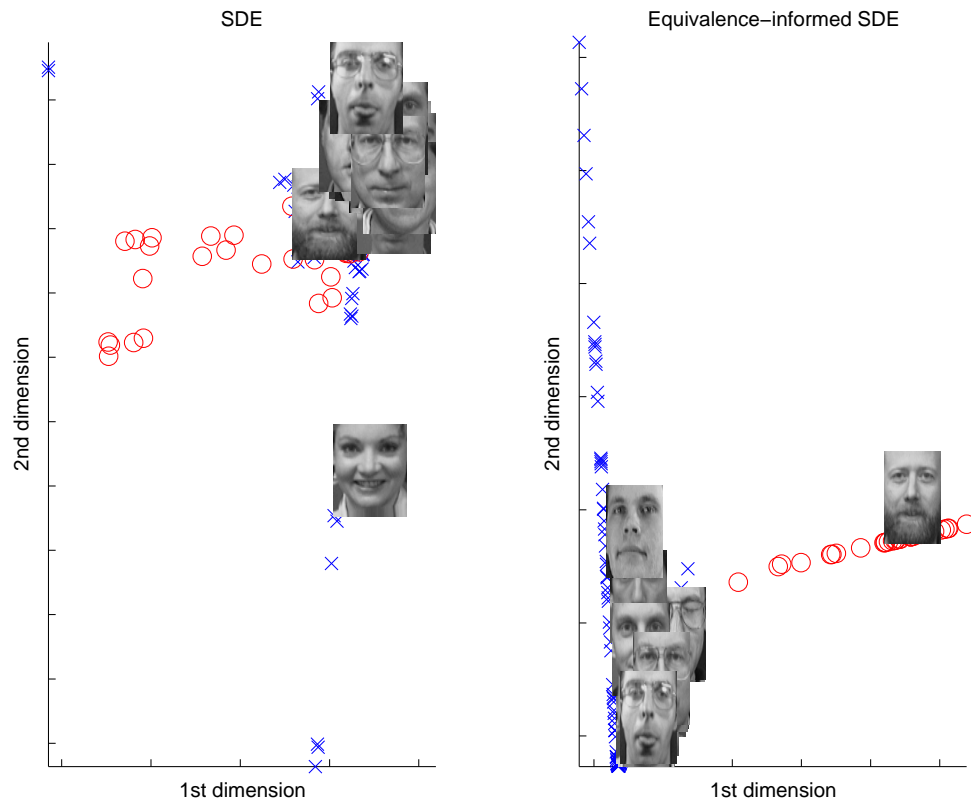


Figure 3.9: *SDE and Equivalence-Informed SDE with bearded/unbearded distinction (all class-equivalent pairs)*

The conclusion to be drawn from these results is that the side information can be effectively exploited to capture characteristics of interest where uninformed embedding techniques fail to automatically discover them. Moreover, two different characteristics (beards and glasses) have been captured within the same data set, even when using only limited quantities of class-equivlance side information. Finally, the preprocessor is effective with a wide-range of embedding techniques.

3.3 Multiple-Attribute Metric Learning with Class-Equivalence Side Information

In some cases, one may have more than one distinction to capture in data (e.g., glasses vs. no glasses **and** male vs. female). The method just presented can be extended to construct a distance metric using multiple sets of side information, each corresponding to a different criterion for similarity. Multiple metrics can be learned, one for each kind of side information, and then be combined to form a new distances metric.

Suppose there are k different sets of side information over the same set of points. Using the optimization described above, k transformations, A_1, \dots, A_k can be learned. From each A_i , the dominant eigenvector v_i can be taken and a new matrix assembled where each column is one of these eigenvectors

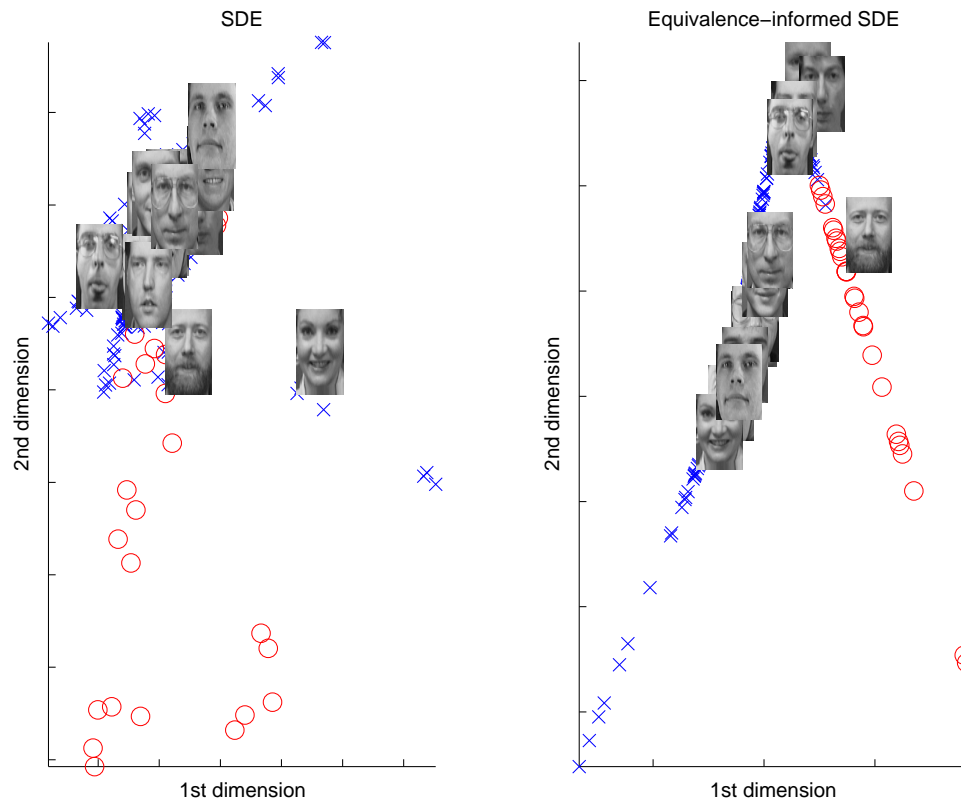


Figure 3.10: *SDE and Equivalence-Informed SDE with bearded/unbearded distinction (four class-equivalent pairs, one class-inequivalent pair)*

$$\bar{A} = \begin{bmatrix} v_1 & \cdots & v_k \end{bmatrix}$$

This combined matrix defines a rank k linear subspace onto which we can project the data. Applying singular value decomposition to \bar{A} , one can form a matrix U from the k eigenvectors corresponding to the largest eigenvalues. The final distance transformation is then $\hat{A} = UU^T$, which is an orthonormal basis combining the distinctions drawn by each kind of side information.

3.3.1 Results

Again, the effectiveness of this approach must be demonstrated, and the methodology for doing so is similar to the previous set of results. Here, however, instead of showing that two characteristics can be discovered individually, the object is to create a single embedding capturing both. A comparison with the uninformed embeddings serves to make the point.

The results shown in Figures 3.11 and A.11-A.14 (in Appendix A) are similar to the earlier experiments but now using both the bearded/unbearded and glasses/no glasses criteria together. Results are shown for all the embedding methods and their multiple-attribute, equivalence-informed versions. In all cases, the side information consisted of all similar pairs and no dissimilar pairs.

The informed embeddings tend to discover the four distinct categories of faces (com-

binations of the presence and absence of beards and glasses). Informed MDS (Figure 3.11) separates these quite well, as do Informed Isomap (Figure A.11) and Informed LLE (Figure A.13). Results with Informed LEM (Figure A.12) are roughly grouped, but only partially separated. Finally, Informed SDE (A.14) does not group or separate very effectively, although some useful structure is present. In all cases, the informed embeddings have distinct structure that the uninformed embeddings fail to capture. The only feature the uninformed embeddings seem to capture is the relative darkness of the images.

These results show that multiple-attribute approach can capture the desired properties in a single embedding, while the uninformed versions fail to capture either of the properties of interest. Again, the method is straightforward to apply and works with a variety of embedding techniques.

3.4 Kernelizing Metric Learning

Not uncommonly, one needs to consider nonlinear transformations of data in order to apply learning algorithms. One efficient method for doing this is via a kernel that computes a similarity measure between any two data points. In this section, I show how a distance metric can be learned in the feature space implied by a kernel, allowing our use of side information to be extended to nonlinear mappings of the data.

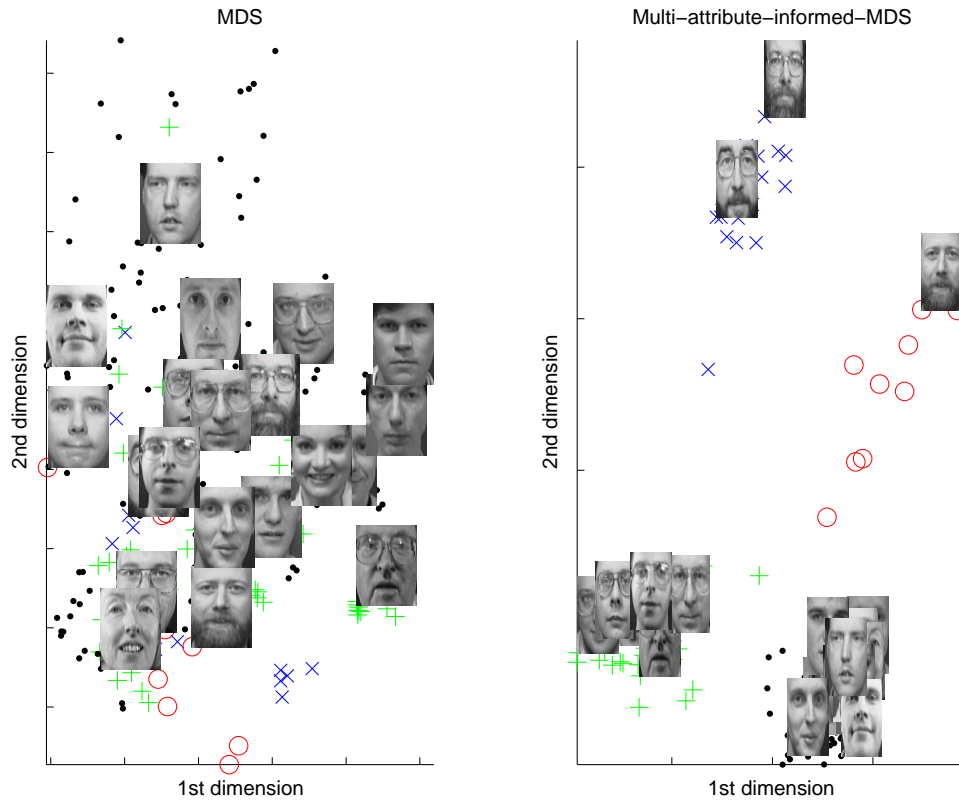


Figure 3.11: *MDS and Multi-Attribute Equivalence-Informed MDS with bearded/unbearded and glasses/no glasses distinctions (all class-equivalent pairs)*

Conceptually, the points are being mapped into a feature space by some nonlinear mapping $\Phi()$ and then a distance metric is learned in that space. Actually performing the mapping is typically undesirable (the feature vectors may have infinite dimension, for example), so I employ the well known *kernel trick*, using some kernel $K(x_i, x_j)$ that computes inner products between feature vectors without explicitly constructing them.

The squared distances in our objective have the form

$$(x_i - x_j)^T A (x_i - x_j)$$

Because A is positive semidefinite, it can be decomposed into $A = WW^T$. This W matrix can then be reexpressed as a linear combination of the data points, $W = X\beta$, via the kernel trick. Rewriting the squared distance

$$\begin{aligned} (x_i - x_j)^T A (x_i - x_j) &= (x_i - x_j)^T WW^T (x_i - x_j) \\ &= (x_i - x_j)^T X\beta\beta^T X^T (x_i - x_j) \\ &= (x_i^T X - x_j^T X)\beta\beta^T (X^T x_i - X^T x_j) \\ &= (X^T x_i - X^T x_j)^T \mathcal{A} (X^T x_i - X^T x_j) \end{aligned}$$

where $\mathcal{A} = \beta\beta^T$, I have now expressed the distance in terms of the matrix to be learned, \mathcal{A} , and inner products between data points, which can be computed via the kernel, K . The optimization of \mathcal{A} then proceeds just as in the non-kernelized version presented earlier, with one additional constraint. The rank of \mathcal{A} must be t because β is t by n and $\mathcal{A} = \beta\beta^T$.

However, this constraint is problematic, and so I drop it during the optimization and then find a rank t approximation of \mathcal{A} using singular value decomposition.

3.5 Using Partial Distance Side Information

I will now discuss the use of the second kind of side information mentioned at the beginning of this chapter. Recall that in this case, we assume that we have prior side information that gives exact distances between some pairs of points in some space natural to the data. This type of side information was motivated by cases where no class structure is available, but some prior distance information can be obtained.

In cases like this, partial distance information can be used to inform our learned metric. Given a set of similarities in the form of pairs for which distances are known

$$S : (x_i, x_j) \in \mathcal{S} \text{ if the target distance } d_{ij} \text{ is known}$$

the following cost function is employed, which attempts to preserve the set of known distances

$$L(A) = \sum_{(x_i, x_j) \in \mathcal{S}} \left| \|x_i - x_j\|_A^2 - d_{ij} \right|^2$$

The optimization problem is then

$$\begin{aligned} & \min_A L(A) \\ & \text{s.t. } A \succeq 0 \end{aligned}$$

A convenient form for this optimization problem is obtained using the same approach for quadratic terms used earlier.

$$\begin{aligned}
L(A) &= \sum_{(x_i, x_j) \in \mathcal{S}} \left\| \|x_i - x_j\|_A^2 - d_{ij} \right\|^2 \\
&= \sum_{(x_i, x_j) \in \mathcal{S}} \left\| \text{vec}(A)^T \text{vec}((x_i - x_j)(x_i - x_j)^T) - d_{ij} \right\|^2 \\
&= \sum_{(x_i, x_j) \in \mathcal{S}} \left\| \text{vec}(A)^T \text{vec}(B_{ij}) - d_{ij} \right\|^2 \\
&= \sum_{(x_i, x_j) \in \mathcal{S}} \text{vec}(A)^T \text{vec}(B_{ij}) \text{vec}(B_{ij})^T \text{vec}(A) + d_{ij}^2 - 2d_{ij} \text{vec}(A)^T \text{vec}(B_{ij})
\end{aligned}$$

where $B_{ij} = (x_i - x_j)(x_i - x_j)^T$. Note that the d_{ij}^2 term in the above is a constant so it can be dropped for the purposes of minimization, and rewrite the loss as

$$\begin{aligned}
L(A) &= \text{vec}(A)^T \left[\sum_{(x_i, x_j) \in \mathcal{S}} \text{vec}(B_{ij}) \text{vec}(B_{ij})^T \text{vec}(A) - 2 \sum_{(x_i, x_j) \in \mathcal{S}} d_{ij} \text{vec}(B_{ij}) \right] \\
&= \text{vec}(A)^T [Q \text{vec}(A) - 2R]
\end{aligned}$$

where $Q = \sum_{(x_i, x_j) \in \mathcal{S}} \text{vec}(B_{ij}) \text{vec}(B_{ij})^T$ and $R = \sum_{(x_i, x_j) \in \mathcal{S}} d_{ij} \text{vec}(B_{ij})$. This objective is still quadratic but a linear objective can be obtained via the Schur complement [6]. I will briefly outline this approach.

The Schur complement relates the positive semidefiniteness of the matrix on the left and the expression on the right by an if-and-only-if relation

$$\begin{bmatrix} X & Y \\ Y^T & Z \end{bmatrix} \succeq 0 \Leftrightarrow Z - Y^T X^{-1} Y \succeq 0$$

By decomposing $Q = S^T S$, a matrix can be constructed of the form

$$J = \begin{bmatrix} I & S\text{vec}(A) \\ (S\text{vec}(A))^T & 2\text{vec}(A)^T R + t \end{bmatrix}$$

and by the Schur complement, if $J \succeq 0$, then the following relation holds

$$2\text{vec}(A)^T R + t - \text{vec}(A)^T S^T S \text{vec}(A) \geq 0$$

Note that the left-hand side of this relation is scalar. So, as long as J is positive semidefinite, the scalar t is an upper bound on the loss

$$\begin{aligned} \text{vec}(A)^T S^T S \text{vec}(A) - 2\text{vec}(A)^T R &= \text{vec}(A)^T Q \text{vec}(A) - 2\text{vec}(A)^T R \\ &\leq t \end{aligned}$$

Therefore, minimizing t subject to $J \succeq 0$ also minimizes the objective. This produces the final optimization problem, which can be readily solved by standard semidefinite programming software

$$\begin{aligned} &\min_A t \\ \text{s.t. } &A \succeq 0 \\ &J \succeq 0 \end{aligned}$$

3.5.1 Results

The final set of results for this chapter follows the same overall methodology. To demonstrate that the limited distance information can inform an embedding regarding some particular characteristic, experiments are run with and without the preprocessing. As with the class-equivalence side information, embeddings structured according to the desired property demonstrate the value of the method.

The embeddings shown in Figures 3.12-3.16 and A.15-A.19 (in Appendix A) use the same face data as the earlier results, but the side information is a set of distances. The distances used here are a portion of distance D^A computed in Section 3.2.2. That is, there are 200 images of faces and two distinctions are identified by class-equivalence side information: faces with beards vs. faces without beards and faces with glasses vs. faces without glasses. For each distinction, the distance matrix D^A is computed where no dissimilar pairs but all similar pairs are identified. Then 30 pairs of points are selected at random and the distance between these pairs is provided from D^A , as the side information for learning a new distance metric. This new distance metric is used then to inform the embeddings.

Results using these distances are somewhat noisier than with the class-equivalence informed embeddings. They still provide good structure, however, compared with the uninformed embeddings, and group the data. Informed MDS (Figures 3.12 and A.15) and Informed Isomap (Figures 3.13 and A.16) group quite effectively, with a few images placed

inappropriately. Informed LEM (Figures 3.14 and A.17), Informed LLE (Figures 3.15 and A.18), and Informed SDE (Figures 3.16 and A.19) all give results similar to one another, and all show noisier performance on the glasses attribute than on the beards (beards are likely easier since they create a simpler and more substantial difference in the image than the subtle distinction of glasses). Among these, Isomap seems to give substantially better results on the glasses while the rest are all rather noisy. All methods group beardedness quite well.

These last results show that preprocessing based on partial distance information is effective, although arguably to a lesser degree than the class-equivalence information. The informed embeddings are still considerably more useful than the uninformed.

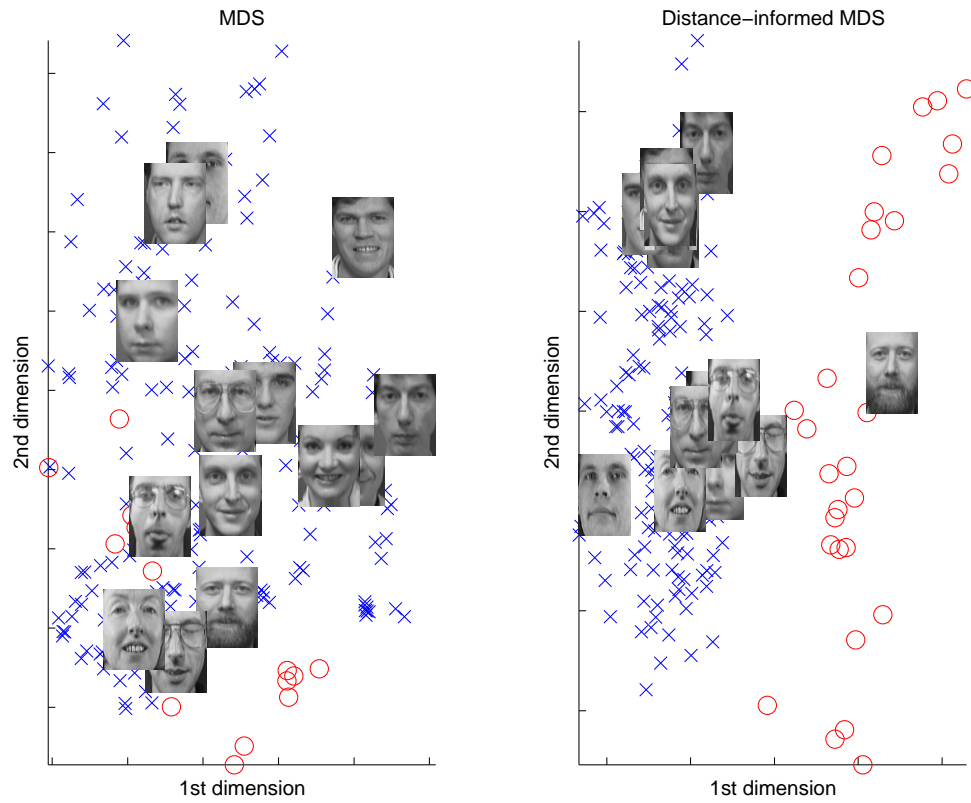


Figure 3.12: *MDS and Distance-Informed MDS with bearded/unbearded distinction (30 distances)*

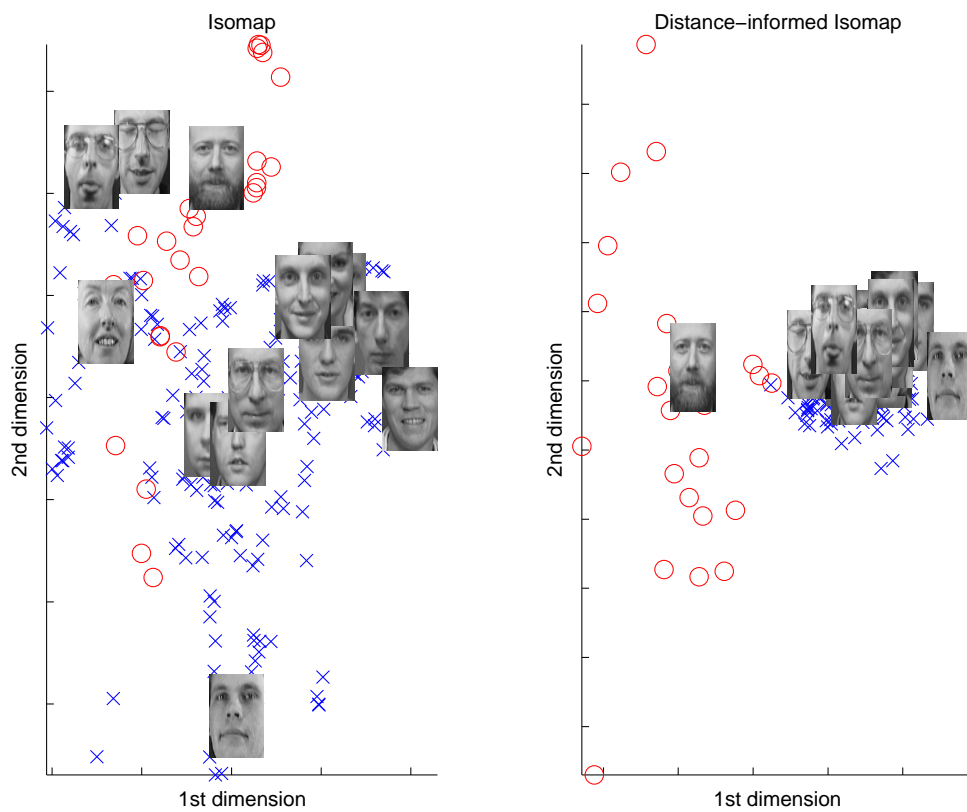


Figure 3.13: *Isomap and Distance-Informed Isomap with bearded/unbearded distinction (30 distances)*

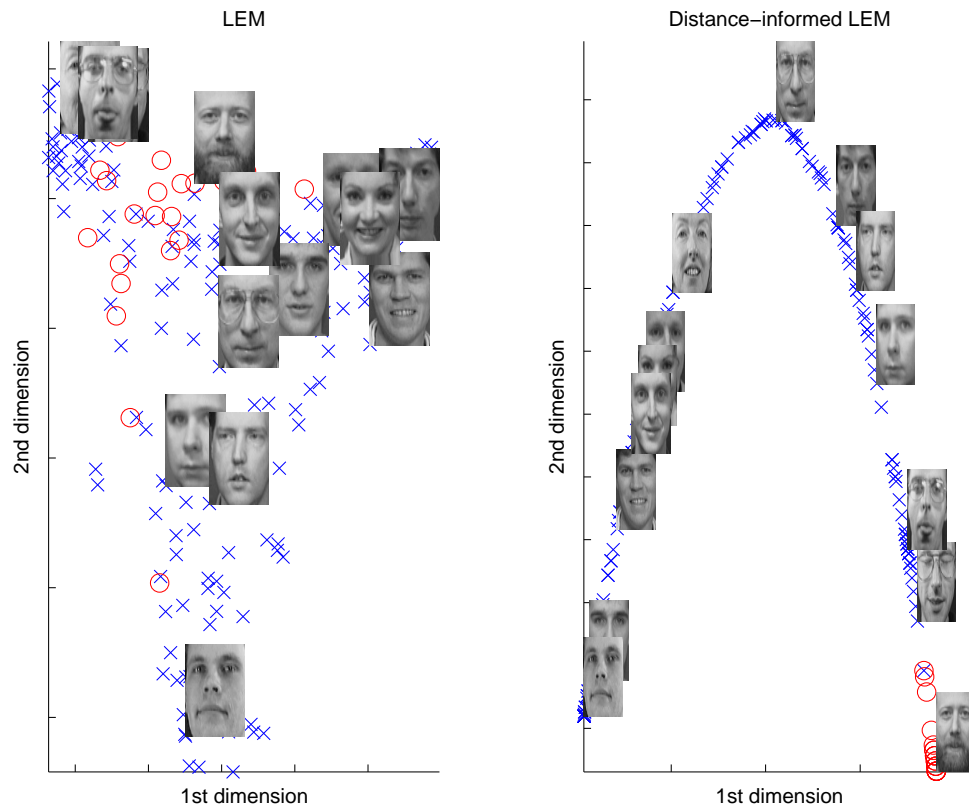


Figure 3.14: *LEM and Distance-Informed LEM with bearded/unbearded distinction (30 distances)*

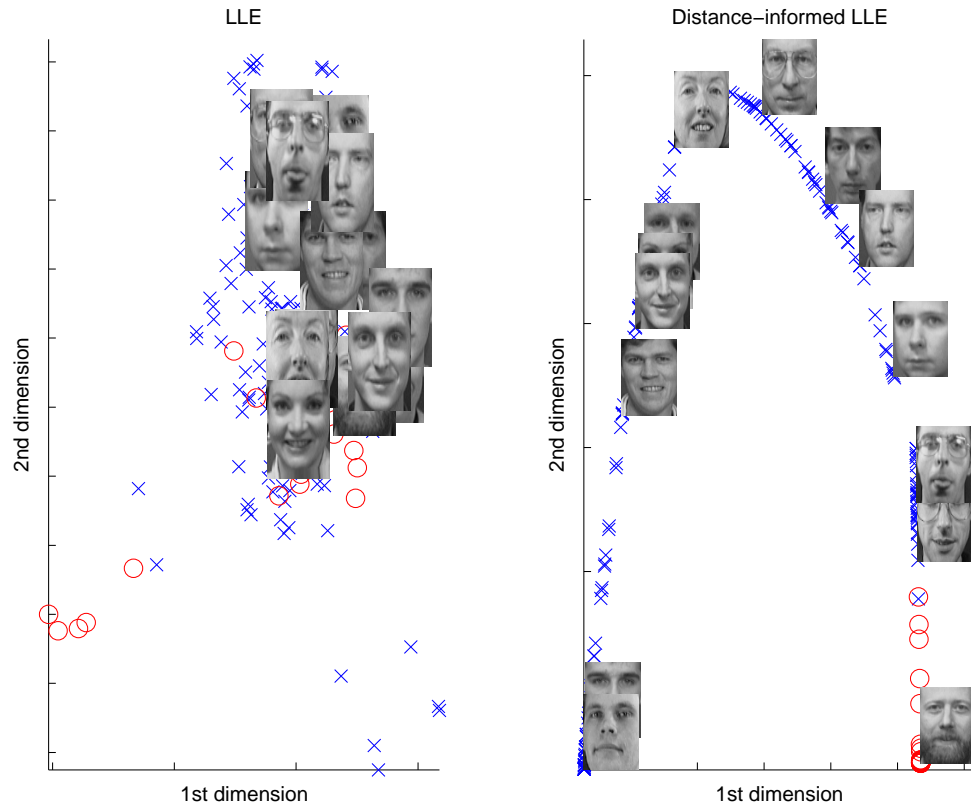


Figure 3.15: *LLE and Distance-Informed LLE with bearded/unbearded distinction (30 distances)*

3.6 Conclusions

A great many machine learning techniques handle complex data by mapping it into new spaces and then applying standard techniques. In many cases, these utilize distances in the new space. Learning a distance metric directly is an attractive simplification of this overall approach, acknowledging that, most of the time, all we really care about are the relative distances. This chapter has shown how side information of two forms, class-equivalence information and partial distance information, can be used to learn a new distance as a preprocessing step for embedding.

The results, shown for a variety of contemporary embedding techniques, demonstrate that the use of side information allows us to capture attributes of the data within the embedding in a controllable manner. Even a small amount of side information can give much better structure to the embedding. Furthermore, the method can be kernelized and also used to capture multiple attributes in the same embedding. Its advantages over existing metric learning methods are a simpler optimization formulation that can be readily solved with off-the-shelf software, and greater flexibility in the nature of side information one can provide. In all, this technique represents a useful addition to our toolbox for informed embeddings.

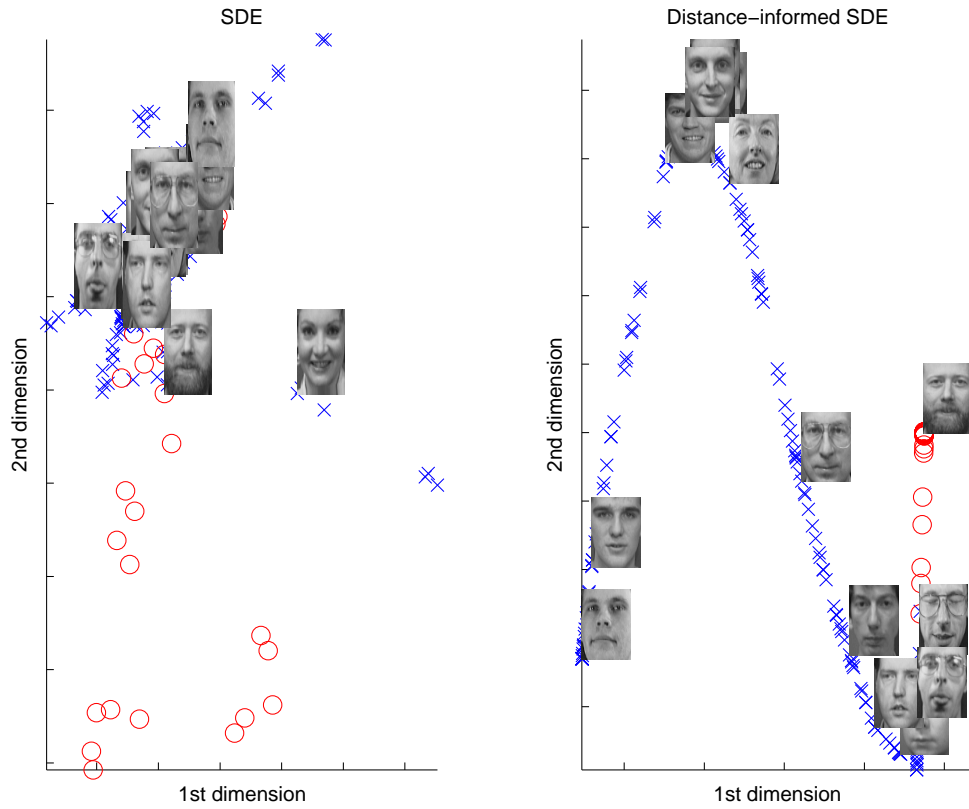


Figure 3.16: *SDE* and *Distance-Informed SDE* with bearded/unbearded distinction (30 distances)

Chapter 4

Action Respecting Embedding

4.1 Introduction

In this chapter I introduce a manifold learning algorithms that potentially can be useful for the application of sequential decision problems¹. I show how this variation on dimensionality reduction can be solved with a semidefinite program and evaluate the technique in a synthetic, robot-inspired domain, demonstrating both qualitatively superior representations, and quantitative improvements on a data prediction task.

Consider sensor readings, such as images, taken from a mobile robot. The most natural representation of the robot's observations would be the robot's pose (e.g., for a wheeled

¹Some of the work reported in this chapter has appeared in [4].

robot: x , y and θ describing the robot’s position and orientation), which allows the high-dimensional image data to be described with only a few dimensions. This representation is desirable not only because it is low-dimensional, but because within it the robot’s actions (e.g., forward and rotation) can be described as simple transformations. This is why the robot’s objective pose is such an ideal representation for robot planning and localization. There is no natural way, though, to encode either the robot’s actions nor the desire that the representation respect these actions through a simple similarity function.

Here, I introduce a new algorithm called Action Respecting Embedding (ARE) to address this variation on traditional manifold learning. Here, the input data are given in sequence, along with uninterpreted² action labels that are associated with adjacent pairs of data points. ARE finds a low-dimensional representation of the input data where the actions are simple transformations in the learned representation. For ARE to extract such a representation it exploits the knowledge of action labels in two key ways:

1. It uses the action-labeled pairs of data points to build a *non-uniform neighbourhood graph*. The graph is constructed using the assumption that pairs of data points that can be reached in a small number of actions should be nearby in the learned repre-

²By uninterpreted we mean that the action labels themselves have no implied meaning. We may refer to actions as being ‘move left’ or ‘move right’ while the algorithm sees the actions as simply ‘Action 1’ and ‘Action 2’.

sensation. Other nonlinear manifold-learning techniques use a k -nearest neighbour graph with a globally uniform k which can create overly dense neighbourhood graphs.

2. The action labels themselves individually have no implied meaning. However, every time an action is repeated it provides more implicit information about the data. From these repetitions we can build *action respecting constraints* that ensure that each action corresponds to a simple transformation in the learned representation.

Using non-uniform neighbourhoods and action respecting constraints, ARE constructs a semidefinite program to learn a kernel that describes the desired low-dimensional representation. The result is a very natural representation of the original high-dimensional data, with a strong correspondence to the actual low-dimensional process that generated the data. Although manifold learning techniques often rely on qualitative evaluation, our knowledge of the actions involved in generating the data allow for a more objective evaluation. Therefore, along with traditional qualitative comparisons I also introduce the task of data prediction as a quantitative measure of the success of our learned representations.

This algorithm is based on semidefinite embedding which has been reviewed in 1.7. The Action Respecting Embedding algorithm is introduced in Section 4.2. Section 4.2.1 shows how non-uniform neighbourhood graph can be extracted. Additional manifold constraints which respect the action labeling are discussed in Section 4.2.2. Section 4.2.3 introduces the task of data prediction and show how ARE can solve this problem. Experimental results

of the proposed algorithm are presented in Section 4.3 before the conclusion in Section 4.4.

4.2 Action Respecting Embedding

Action respecting embedding takes a sequence of high-dimensional data x_1, \dots, x_t , along with associated discrete actions a_1, \dots, a_{t-1} . The data are assumed to be in some order, where action a_i was taken between data points x_i and x_{i+1} . The final piece of input is a similarity function, $\|x_i - x_j\|$, defining a distance over the high-dimensional data points. For vector data, Euclidean distance is often sufficient, but other data-specific similarities can be employed.

The overall structure of the algorithm follows the same three steps of SDE (see 1.7):

- (i) Construct a neighbourhood graph.
- (ii) Solve a semidefinite program to find the maximum variance embedding subject to constraints.
- (iii) Extract a low-dimensional embedding from the dominant eigenvectors of the learned kernel matrix.

ARE, though, seeks to exploit the additional information provided by the action labels of the data. This information is exploited through two key modifications. The first modifies step (i) by constructing non-uniform neighbourhoods based on action-labeled pairs of data

points. The second modifies step (ii) by adding action-respecting constraints into the semidefinite program.

4.2.1 Non-Uniform Neighbourhoods

Many of the current nonlinear manifold learning techniques seek to preserve local properties of the original data. They often require a neighbourhood graph over the original data points to define a notion of locality. As we have seen, SDE creates this graph by connecting each data point to its k -nearest neighbours for some chosen value of k . Since the neighbourhood graph must be fully connected for SDE to have a bounded solution, this choice of k can be forced to be quite large and may over-constrain the learned manifold. Another possibility would be to choose a distance threshold ϵ and connect any two data points within that threshold as neighbours. Again, this may result in an over-constrained manifold as ϵ must be set large enough to make the graph fully connected. The key drawback in these techniques is that they require a globally uniform k or ϵ .

Since we are given additional information relating the points in our set, i.e., that certain pairs of data points are connected by an action, we can build a more intuitive, non-uniform neighbourhood graph. The idea is based on the assumption that data points connected by an action are nearby and should be considered neighbours. We can use these assumed neighbours to define a neighbourhood ball around each data point, whose radius is large

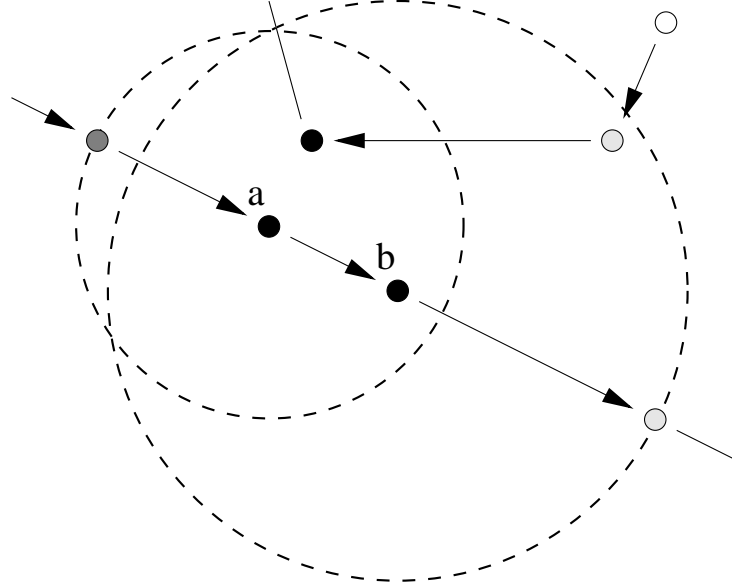


Figure 4.1: *An example of the use of action labels to find non-uniform neighbourhoods. The arrows show the points that are connected by an action. The circles show the resulting neighbourhood for the points labeled ‘a’ and ‘b’ with $T = 1$. Black points are in both neighbourhoods. White points in neither. Shaded points are in one but not the other.*

enough to encompass all data points connected by an action. We can then include an edge in the neighbourhood graph between two images if they are both in each other’s neighbourhood ball. The connectivity of the neighbourhood graph can be increased by increasing the action window, i.e., requiring data points within T actions of each other to be neighbours. Since our data is generated from a sequence of actions, we can define the neighbourhood graph as follows. Let η_{ij} be the adjacency matrix of the neighbourhood

graph. Given an action window of T ,

$$\begin{aligned}
 \eta_{ij} = 1 &\Leftrightarrow \exists k, l \text{ such that} \\
 |k - i| < T, \quad |l - j| < T, \\
 \|x_i - x_k\| &> \|x_i - x_j\| \quad \text{and} \\
 \|x_j - x_l\| &> \|x_i - x_j\|.
 \end{aligned} \tag{4.1}$$

Figure 4.1 shows an example of two-dimensional data points connected by actions, and the resulting neighbourhood balls when $T = 1$.

Notice that since our data points come from a sequence of actions, the resulting neighbourhood graph ($T \geq 1$) must be fully connected. This satisfies the critical requirement that the semidefinite optimization be bounded (otherwise a solution would not exist).

4.2.2 Action Respecting Constraints

The second, and most important, contribution of ARE is the addition of action respecting constraints. The evaluation of learned manifolds is often subjective and usually amounts to demonstrating that the manifold corresponds to the known data generator's own underlying degrees of freedom. Action labels, even with no interpretation or implied meaning, provide more information about the underlying generation of the data. It is natural to expect that the actions correspond to some simple operator on the generator's own degrees of freedom.

For example, a camera that is being panned left and then right, has actions that correspond to a simple translation in the camera's actuator space. We therefore want to constrain the learned representation so that the labeled actions correspond to simple transformations in that space. In particular, we can require all actions to be a simple rotation plus translation in the resulting low-dimensional representation.³

We can formalize this constraint by first observing rotation plus translation is exactly the space of *distance preserving* transformations. A transformation \mathcal{T} is distance preserving and thus a rotation plus translation if and only if:

$$\forall x, x' \quad ||\mathcal{T}(x) - \mathcal{T}(x')|| = ||x - x'||.$$

Let us consider this in the context of an action-labeled data sequence. All actions must be distance preserving transformations in the learned representation. Therefore, for any two data points, x_i and x_j , the same action taken at those data points must preserve their distance. Letting $\Phi(x_i)$ denote data point x_i in the learned space, We require that action a 's transformation, \mathcal{T}_a , must satisfy:

$$\begin{aligned} \forall i, j \quad ||\mathcal{T}_a(\Phi(x_i)) - \mathcal{T}_a(\Phi(x_j))|| = \\ ||\Phi(x_i) - \Phi(x_j)||. \end{aligned} \tag{4.2}$$

Now, if we let $a = a_i$ and consider the case where $a_j = a_i$. Then, $\mathcal{T}_a(\Phi(x_i)) = \Phi(x_{i+1})$ and

³These are the subset of linear transformations that don't involve any scaling component.

$\mathcal{T}_a(\Phi(x_j)) = \Phi(x_{j+1})$, and Constraint (4.2) becomes:

$$||\Phi(x_{i+1}) - \Phi(x_{j+1})|| = ||\Phi(x_i) - \Phi(x_j)||. \quad (4.3)$$

We want to pose this not as a constraint on distances, but rather as a constraint on inner products, i.e., on the learned kernel matrix, K . We can square both sides of the equation and rewrite it in terms of K resulting in the following set of constraints:

$$\begin{aligned} \forall i, j \quad a_i = a_j \Rightarrow \\ K_{(i+1)(i+1)} - 2K_{(i+1)(j+1)} + K_{(j+1)(j+1)} = \\ K_{ii} - 2K_{ij} + K_{jj} \end{aligned} \quad (4.4)$$

We can add Constraint (4.4) into SDE's usual constraints to arrive at the optimization and algorithm shown in Table 4.1. There is a slight modification to SDE's usual neighbour constraint, changing strict equality into an upper bound. This modification insures that the constraints are feasible by allowing the zero matrix to be a feasible solution. Notice that the additional action respecting constraints are still linear in the optimization variables, K_{ij} , and so the optimization remains a semidefinite program. Since the neighbourhood graph η_{ij} is fully connected, the optimization is bounded, convex, and feasible, and therefore can be solved efficiently with various general-purpose toolboxes. The results in this paper were obtained using SeDuMi [28] in MATLAB. These results also used highly penalized slack variables in SDE's neighbourhood constraint to help improve the stability of the solution.

This was recommended by Weinberger and colleagues in their original SDE work [35].

Algorithm: ARE

Construct neighbours, η , according to Equation (4.1).

Maximize $\text{Tr}(K)$ subject to $K \succeq 0$: $\sum_{ij} K_{ij} = 0$,

$$\forall ij \quad \eta_{ij} > 0 \vee [\eta^T \eta]_{ij} > 0 \Rightarrow$$

$$K_{ii} - 2K_{ij} + K_{jj} \leq ||x_i - x_j||^2 \quad , \text{ and}$$

$$\forall ij \quad a_i = a_j \Rightarrow$$

$$K_{(i+1)(i+1)} - 2K_{(i+1)(j+1)} + K_{(j+1)(j+1)} =$$

$$K_{ii} - 2K_{ij} + K_{jj}$$

Run Kernel PCA with learned kernel, K .

Table 4.1: ARE Algorithm.

4.2.3 Data Prediction

As manifold learning is an unsupervised learning problem, evaluation of algorithms is usually qualitative. Data prediction, though, is a task I introduce, which (i) can be measured quantitatively and (ii) seeks to evaluate how well a low-dimensional representation has captured the actions. The problem of data prediction is: given a data point and an action, predict the resulting data point. In general, this is a very challenging task. Manifolds

learned with ARE can be used to tackle a partial version of this task: given a data point and action from the training set, x_i and a (where a is not necessarily a_i), predict the next data point assuming it is also a data point from the training set. Here I describe how ARE can be used to solve this task, and in Section 4.3 I present the results of this quantitative evaluation of the accuracy of ARE's predictions.

ARE learns an embedding where actions correspond to distance preserving transformations. Once we have obtained an embedding, we can recover the transformations corresponding to the actions and use them for prediction. Recall Constraint (4.2), this implies:

$$\forall i, j \quad ||\mathcal{T}_a(\Phi(x_i)) - \mathcal{T}_a(\Phi(x_j))|| = ||\Phi(x_i) - \Phi(x_j)||.$$

Considering only j 's such that $a_j = a$, results in the following constraint on the result of the action's transformation:

$$\begin{aligned} \forall j \quad a_j = a \Rightarrow \quad & ||\mathcal{T}_a(\Phi(x_i)) - \Phi(x_{j+1})|| = \\ & ||\Phi(x_i) - \Phi(x_j)||. \end{aligned} \tag{4.5}$$

If action a appeared in our training set m times, then this gives us m constraints on $\mathcal{T}_a(\Phi(x_i))$'s distance to other known points, $\Phi(x_{j+1})$. In fact, if the learned manifold has dimensionality d , $d + 1$ independent distance constraints uniquely determines $\mathcal{T}_a(\Phi(x_i))$. In this case, it is a simple matter to find the point $\Phi(x_p)$ nearest to the constrained point $\mathcal{T}_a(\Phi(x_i))$, and use x_p as our prediction. If the point is under-constrained ($m \leq d$), then

we can select the index p according to:

$$p = \operatorname{argmax}_{k=1\dots t} \sum_{j:a_j=a} \left(\frac{\|\mathcal{T}_a(\Phi(x_i)) - \Phi(x_{j+1})\| - \|\Phi(x_k) - \Phi(x_{j+1})\|}{\|\Phi(x_k) - \Phi(x_{j+1})\|} \right)^2. \quad (4.6)$$

In other words, $\Phi(x_p)$ is the embedded point whose distances to other points most closely agrees with $\mathcal{T}_a(\Phi(x_i))$'s distance constraints. We can then use x_p as our prediction.

4.3 Results

I now examine the effect of ARE's non-uniform neighbourhoods and action respecting constraints on learning low-dimensional action-respecting representations. These results are in a synthetic, robot-inspired, image manipulation domain called IMAGEBOT. I first present this domain and then show manifolds produced by ARE and SDE from data generated in this domain.⁴ In addition to the qualitative comparisons, I also present a quantitative evaluation using the data prediction task described in Section 4.2.3

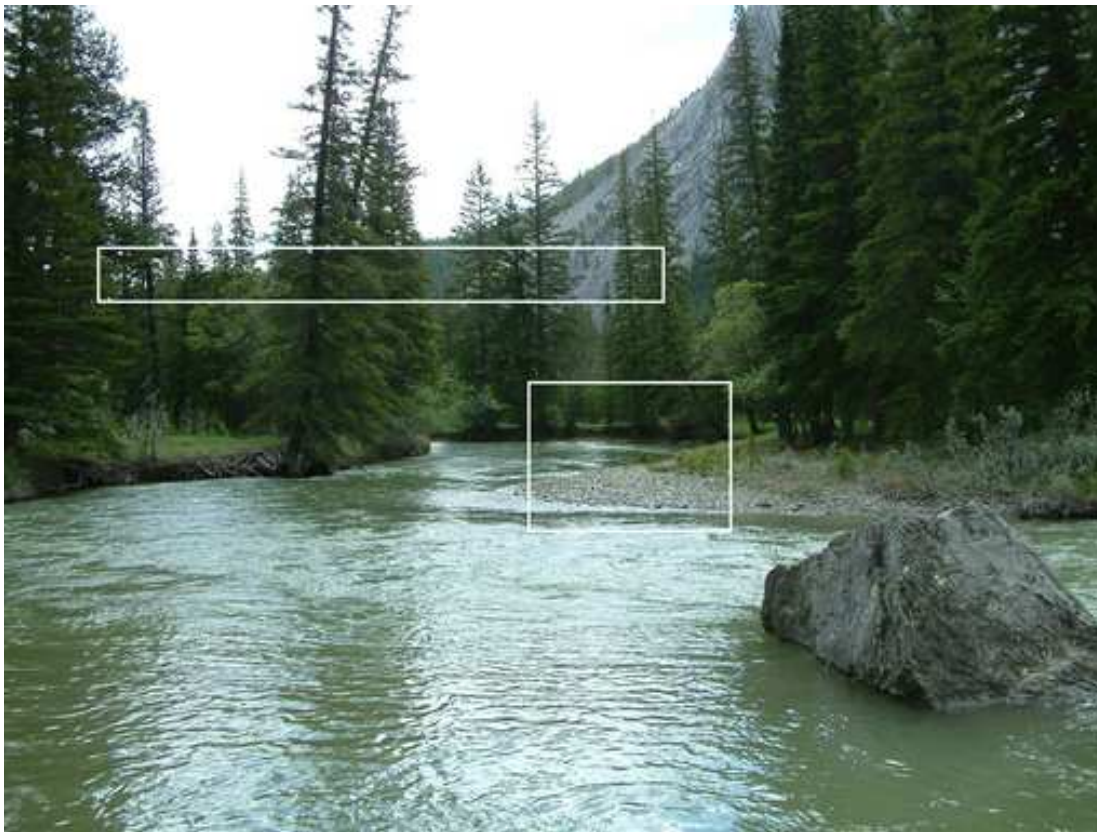


Figure 4.2: IMAGEBOT’s world.

4.3.1 IMAGEBOT Domain

Given an image, one can imagine a virtual robot that can observe a small patch on that image and also take actions to move the observable patch around the larger image. This

⁴ The results from SDE are provided to demonstrate the improvement that can be gained by proper exploitation of action labels. While SDE is quite competent at standard manifold learning, it is not expected that other standard techniques perform as well as ARE at this variant of the problem. However, the results of other standard techniques are illustrated in Appendix 1



Figure 4.3: *A sample 60-action trajectory from IMAGEBOT.*

“image robot” provides us with an excellent domain in which we can test ARE, while having obvious connections to robotic applications.

For these experiments, IMAGEBOT is always viewing a 100 by 100 patch of a 2048 by 1536 image. IMAGEBOT is restricted to eight distinct actions: four translation actions, two rotation actions and two zoom actions. The translations are ‘forward’, ‘backward’, ‘left’ and ‘right’, each by 25 pixels. The rotation actions are ‘turn left’ and ‘turn right’, each by $22\frac{1}{2}^\circ$. The zoom actions are ‘zoom in’ and ‘zoom out’, each changing the scale by a factor of $\sqrt[8]{2}$ (i.e., eight zoom actions double the image scale).

Figure 4.2 shows the image used for the experiments, while Figure 4.3 shows an example trajectory from IMAGEBOT (Figure 4.3 is an enlargement of the long, thin highlighted rectangular section in Figure 4.2.) The trajectory starts on the far left with IMAGEBOT facing right. IMAGEBOT then takes 40 steps forward (to the right) and then 20 steps backward. Figure 4.4 shows a more complicated ‘A’-shaped trajectory that IMAGEBOT followed (Figure 4.4 is a blow up of the other highlighted rectangular section in Figure 4.2.)

IMAGEBOT’s observations as it follows these paths, along with the actions associated

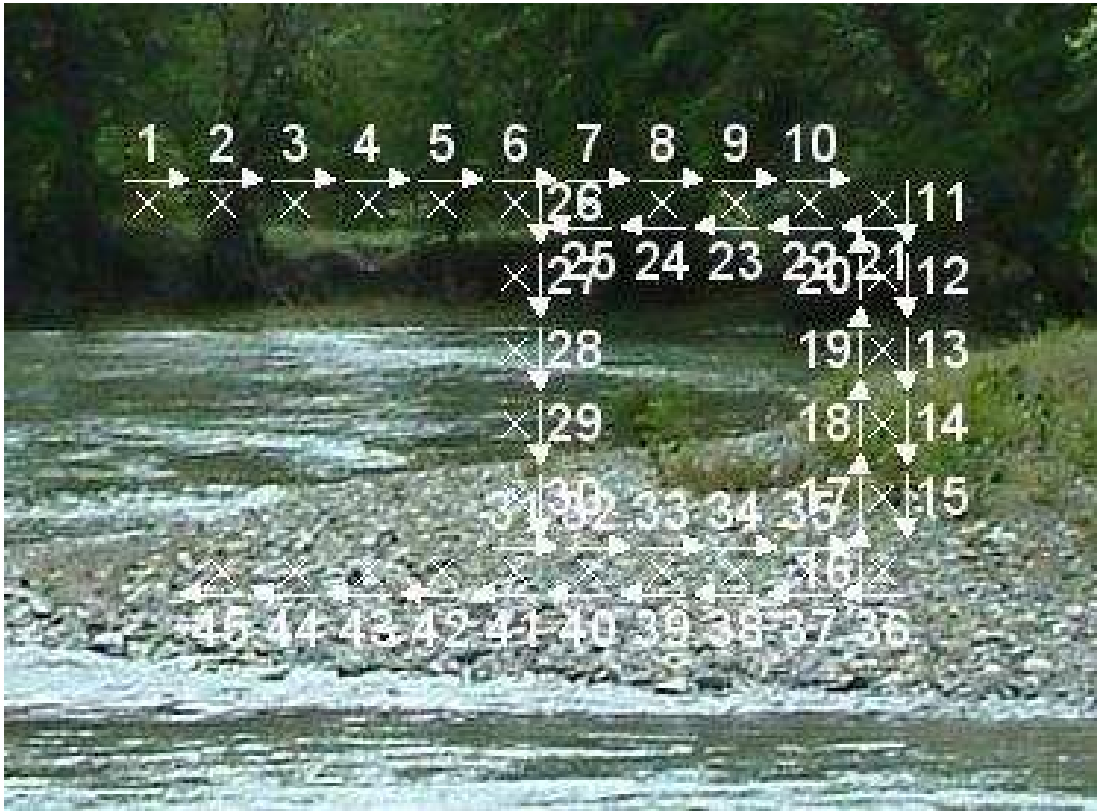


Figure 4.4: A more complicated 45-action trajectory from IMAGEBOT

with the paths, gives a perfect domain for testing ARE— ordered high-dimensional data, with each consecutive pair related by an action. Note that while IMAGEBOT knows what action it takes at every step there is no semantic information associated with that knowledge, i.e., the labels are uninterpreted.

4.3.2 Manifold Learning

Both SDE and ARE were applied to the IMAGEBOT data from the trajectory in Figure 4.3. As might be expected, the resulting manifold for both algorithms is not surprising—essentially one-dimensional as the first eigenvalue of the resulting kernel dominates the others. Of interest, however, is a plot of the trajectory on this manifold over time, which is shown in Figure 4.5. Note that the result from SDE indicates that IMAGEBOT doubled back on itself seven times. The result from ARE is markedly smoother and corresponds almost exactly to IMAGEBOT’s actual manifold. Despite not having any meaning attached to the actions, ARE has clearly managed to learn a representation which captures the essential properties of the actual actions. Namely, that the two different actions are opposites of each other in terms of direction and have the same magnitude.

We can subtly change the actions which generate the data, making the backward action move twice as far as the forward one. Figure 4.6 demonstrates that ARE is capable of learning a manifold that can capture this property as well.

ARE can correctly handle periodic actions, such as rotation, as well. Figure 4.7 shows the first two dimensions of the manifold corresponding to the trajectory consisting of sixteen ‘turn right’ and eight ‘turn left’ actions. ARE again captures that the two actions are opposites, and that they are periodic.

ARE continues to yield good results in the face of more complicated collections of trans-

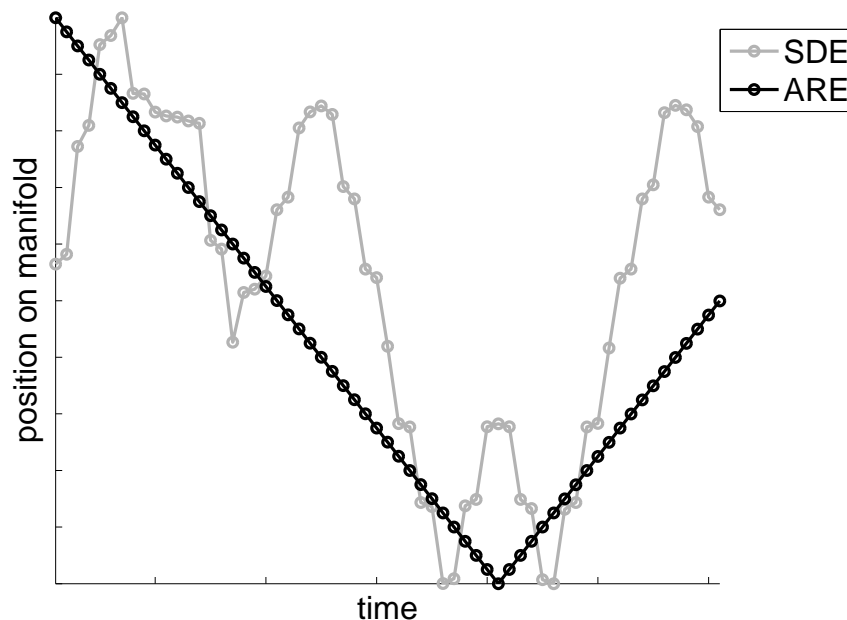


Figure 4.5: *Manifolds from trajectory shown in Figure 4.3. Lines show the distance along the manifold over time.*

formations. ARE and SDE were run with the more complex example shown in Figure 4.4, and the resulting manifolds are displayed in Figure 4.8. SDE, as with the previous example, fails to generate a manifold in which the actions have a simple interpretation. Notice that again, ARE’s manifold has a strong correspondence with IMAGEBOT’s actual trajectory. It again captures the expected relationships between the actions corresponding to forward and back, as well as the actions corresponding to right and left. Even more impressive, the manifold has captured that the forward/back actions are independent and orthogonal to the right/left actions—despite the fact that none of this meaning was explicitly coded

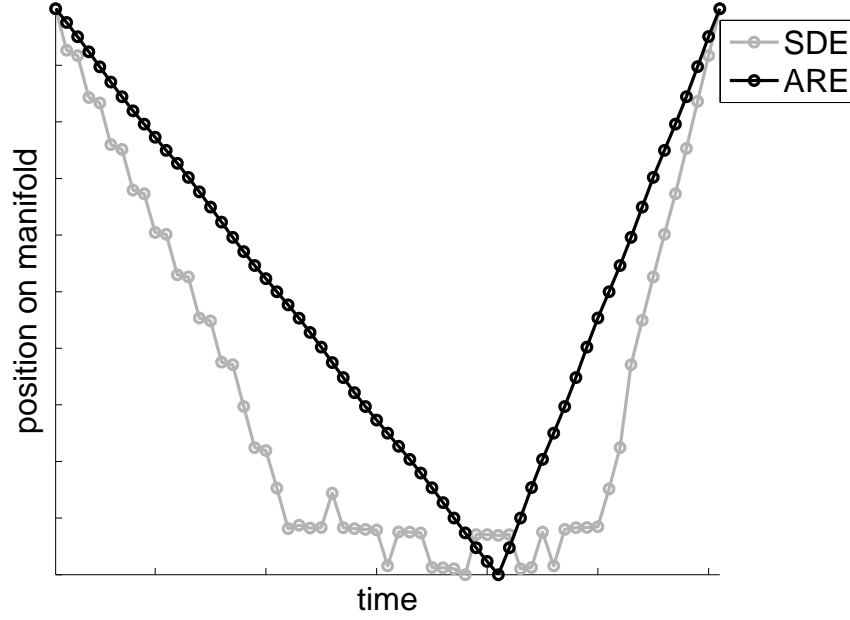


Figure 4.6: *Manifolds from a trajectory similar to that from Figure 4.3 but with slightly different actions. Lines show the distance along the manifold over time.*

in the problem input.

In the final example, IMAGEBOT follows a variation of the ‘A’ trajectory. Instead of the actions ‘left’, ‘right’, ‘forward’ and ‘backward’ IMAGEBOT uses the actions ‘zoom in’, ‘zoom out’, ‘forward’ and ‘backward’. In this case it is no longer true that the two pairs of actions—‘forward’/‘backward’ and ‘zoom in’/‘zoom out’—are independent, as the distance IMAGEBOT moves when implementing the first pair is dependent on IMAGEBOT’s zoom level. Nonetheless, as Figure 4.9 demonstrates, ARE again learns a manifold that captures this relationship. The left leg of the ‘A’ corresponds to images gathered when IMAGEBOT

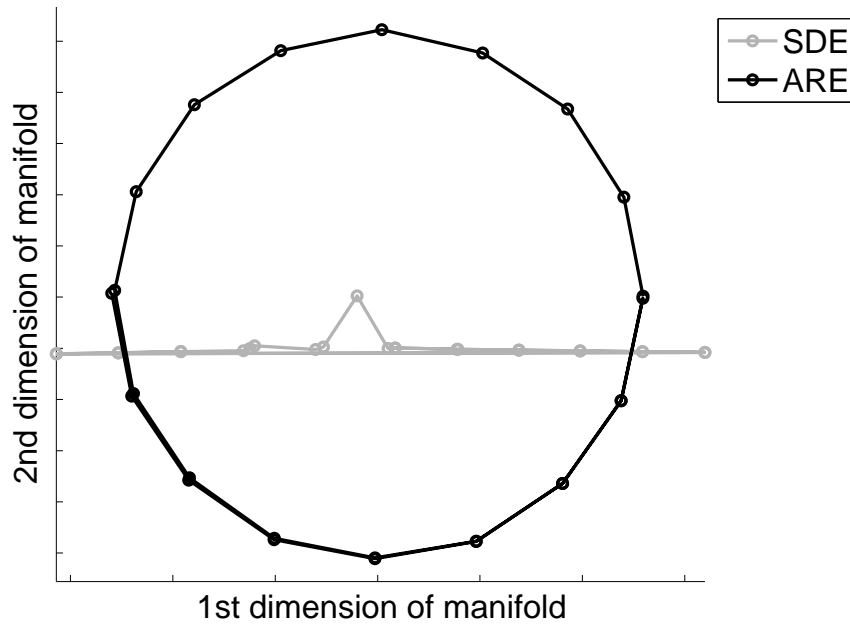


Figure 4.7: *Manifolds learned on data generated with rotation actions.*

was zoomed in, the right leg corresponds to images gathered when IMAGEBOT was zoomed out. Note that distance between consecutive points is less on the left leg than on the right. With this example ARE has successfully learned the radial relationship between the two sets of actions without requiring that the relationship be explicitly known ahead of time.

Finally, ARE is flexible in the choice of image similarity function. All though not shown here, similar results can be obtained using other distance metrics instead of Euclidean distance.

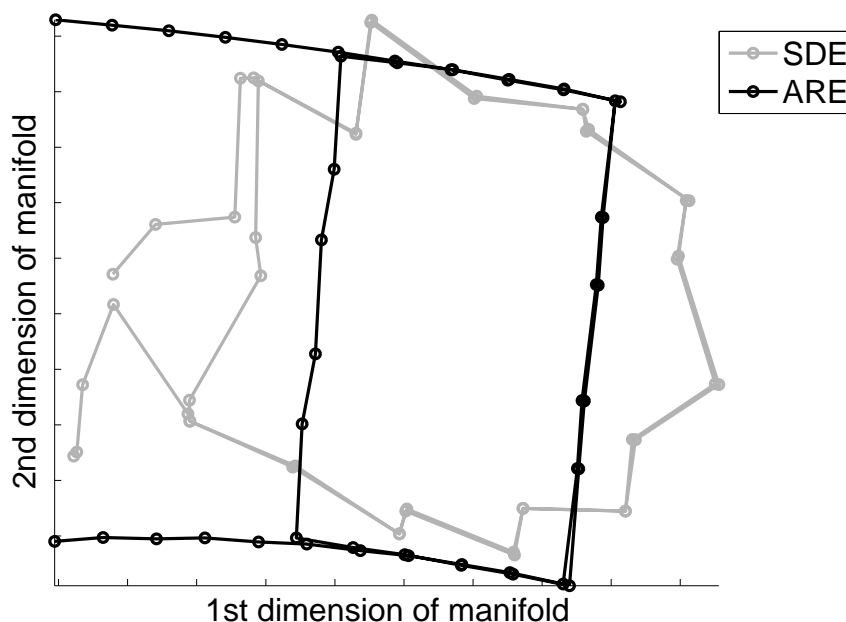


Figure 4.8: *Manifolds corresponding to Figure 4.4*

4.3.3 Data Prediction

In Section 4.2.3 I introduced the task of data prediction, and described how ARE could be used to solve this problem. I applied the data prediction algorithm to the four trajectories from the previous section. As data prediction is a form of supervised learning, we want to be careful to measure accuracy only on queries outside of the training data. Queries of the form, “what training image would result from taking action a_1 from image x_1 ?”, can be answered (x_2) easily from ARE’s input data. Other queries, such as, “in Figure 4.4, what training image would result from taking action a_{11} from image x_{28} ?”, are not so easily answered. This query can only be answered by understanding that some actions are

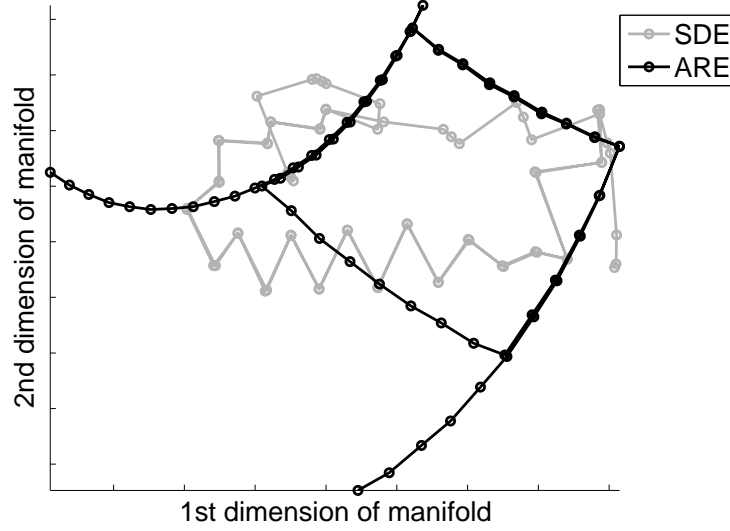


Figure 4.9: *Manifolds learned on data generated with zoom actions.*

inverses of each other, i.e., that the manifold extracts a representation that appropriately respects the action labels.

All possible image-action pairs are generated whose result is an image in the training data. I then excluded all pairs of the form (x_i, a_i) , as these are queries answered directly in the training data. The remaining queries were used to evaluate ARE’s data prediction algorithm. For a comparison baseline, I also performed the same evaluation using manifolds extracted with SDE. To be as accommodating as possible, two prediction techniques for SDE are examined. First, I used ARE’s data prediction algorithm with SDE’s learned manifold. Second, I used regression on SDE’s representation to find the best linear transformation for each action, with the nearest training point to the transformed query point

	Fig. 5	Fig. 6	Fig. 7	Fig. 8	Fig. 9
ARE	100.0%	100.0%	100.0%	100.0%	97.2%
SDE-d	10.2%	14.0%	28.0%	41.7%	25.0%
SDE-l	11.9%	29.8%	20.0%	39.6%	27.8%

Table 4.2: Prediction accuracy across the four trajectories.

being the prediction.

Table 4.2 shows the prediction accuracy for all three methods across the five trajectories. In the table, “SDE-d” refers to SDE using ARE’s data prediction, and “SDE-l” refers to using a linear transformation. ARE achieves near-perfect accuracy, quantitatively demonstrating ARE’s ability to learn better manifolds.

4.4 Conclusions

In summary, I described a variant of standard dimensionality reduction where we are given action labels in addition to data points. Assuming that these labels correspond to particular movements of a camera or other actuator, the goal becomes learning a manifold in which the actions have a meaningful representation.

Although traditional dimensionality reduction methods can be applied to this problem, none of them make effective use of the action labels. I therefore developed ARE—a

semidefinite optimization for solving this problem inspired by SDE. ARE introduces two new critical components. First, ARE uses the action labels to build a non-uniform neighbourhood graph. Second, and more important, we can use these action labels to build constraints which force the learned manifold to be one in which the actions can be represented as simple transformations.

The effectiveness of ARE in learning manifolds from the IMAGEBOT domain are demonstrated, and the results qualitatively and quantitatively are illustrated. ARE was able to capture properties of the actions underlying the original data, despite the fact that none of these properties were explicitly coded in the input. Additionally, ARE greatly outperformed SDE in the provided data-prediction task.

As mentioned in the introduction, low-dimensional representations where actions can be defined as simple transformations are the foundation for many AI applications. Finding a sequence of actions to achieve a particular outcome (i.e., planning) and maintaining a representation of one’s location (i.e., localization) are two such tasks. I have demonstrated that ARE can *automatically* extract representations especially suited to these tasks from only a stream of experience. Although beyond the scope of this chapter, I have successfully co-implemented planning[36] and localization[5] with ARE on small problems. I expect that other standard AI tasks may be able to benefit from ARE’s ability to automatically extract good representations.

Chapter 5

Conclusion

In this thesis, I have introduced three novel approaches to exploit and incorporate side information in the process of manifold discovery.

In Chapter 2, I have presented a dimensionality reduction technique that respects two different similarity measures simultaneously. This can be seen as a hybrid method which combines two different embedding techniques (e.g., combining LLE and Isomap, or Laplacian eigenmaps and MDS, etc.) with a *closed form solution* for the combined cost function.

This chapter demonstrates how we can incorporate valuable side information derived from transformation invariants on images. I have presented two different ways of incorporating transformation invariants in order to make new similarity measures. Known transformations inherent in the image domain can be used to augment nonlinear embed-

ding techniques by correcting potentially misleading Euclidean distances. Transformation corrected embeddings reliably capture the dimension corresponding to a sequence of such transformations and other distinguishing features along with it. Computationally, the method retains the advantages of the techniques we augment by having a closed form solution for the optimization and represents a general purpose method for combining distance-preserving and locality-preserving embedding methods. Future work consists in extending the approach to use multiple transformations simultaneously, more sophisticated methods for characterizing the local manifold, and trying new combinations of embedding techniques.

Chapter 3 takes a more direct approach. It provides the embedding technique with some “hint” regarding a property of interest in the form of side information—limited information about the relationships between points. This chapter shows how two kinds of such side information can be used in a preprocessing step for embedding techniques, leading to embeddings that capture the target properties. The first kind of side information conveys information regarding the classes to which the data belongs, identifying pairs of points that belong to the same class or pairs that belong to different classes. The second kind of side information takes the form of partial information about the similarity of points in the form of distances. It may be that available side information in a domain extends beyond the simple, boolean class-equivalence relation. One may have actual distances available for

some pairs of points, obtained by some expensive measurement or derived as special cases from the nature of the domain.

Chapter 3 shows how side information of two forms, class-equivalence information and partial distance information, can be used to learn a new distance as a preprocessing step for embedding.

The results, shown for a variety of contemporary embedding techniques, demonstrate that the use of side information allows us to capture attributes of the data within the embedding in a controllable manner. Even a small amount of side information can give much better structure to the embedding. Furthermore, the method can be kernelized and also used to capture multiple attributes in the same embedding. Its advantages over existing metric learning methods are a simpler optimization formulation that can be readily solved with off-the-shelf software, and greater flexibility in the nature of side information one can provide. In all, this technique represents a useful addition to our toolbox for informed embeddings.

In Chapter 4, I have considered another form of side information present in domains where the data comes as the result of a sequence of discrete actions. A large number of problems of scientific interest, including the control of robots, industrial processes and inventory management, are sequential decision problems. In these problems, it is often the case that observations are given in a sequence along with actions that relate adjacent

pairs of data points. Therefore, additional information is known about the data, i.e., the sequence of the observations and the actions between data points, but this information cannot be easily represented in the form of similarities. In this chapter, I have presented an algorithm that exploits this additional knowledge and greatly enhances manifold discovery. It is a variation on dimensionality reduction, where the output is a representation of the input data that is both low-dimensional and respects the actions (i.e., actions correspond to simple transformations in the output representation). The idea is to estimate a kernel matrix that implicitly maps the original data space into a feature space, automatically discovering a mapping that unfolds the underlying manifold such that the actions correspond to simple transformations of points on the learned manifold. The computational method involves optimizing a positive semidefinite matrix by maximizing the variance in the feature space, subject to constraints that preserve the distances between nearest neighbours and consistency constraints derived from the action labels that relate observations. In summary, I have described a variant of standard dimensionality reduction where we are given action labels in addition to data points. Assuming that these labels correspond to particular movements of a camera or other actuator, the goal becomes learning a manifold in which the actions have a meaningful representation.

Although traditional dimensionality reduction methods can be applied to this problem, none of them make effective use of the action labels.

Low-dimensional representations where actions can be defined as simple transformations are the foundation for many AI applications. Finding a sequence of actions to achieve a particular outcome (i.e., planning) and maintaining a representation of one's location (i.e., localization) are two such tasks. I have demonstrated that ARE can *automatically* extract representations especially suited to these tasks from only a stream of experience.

The common theme of these techniques is incorporating side information that usually ignored by existing dimensionality reduction methods.

Appendix A

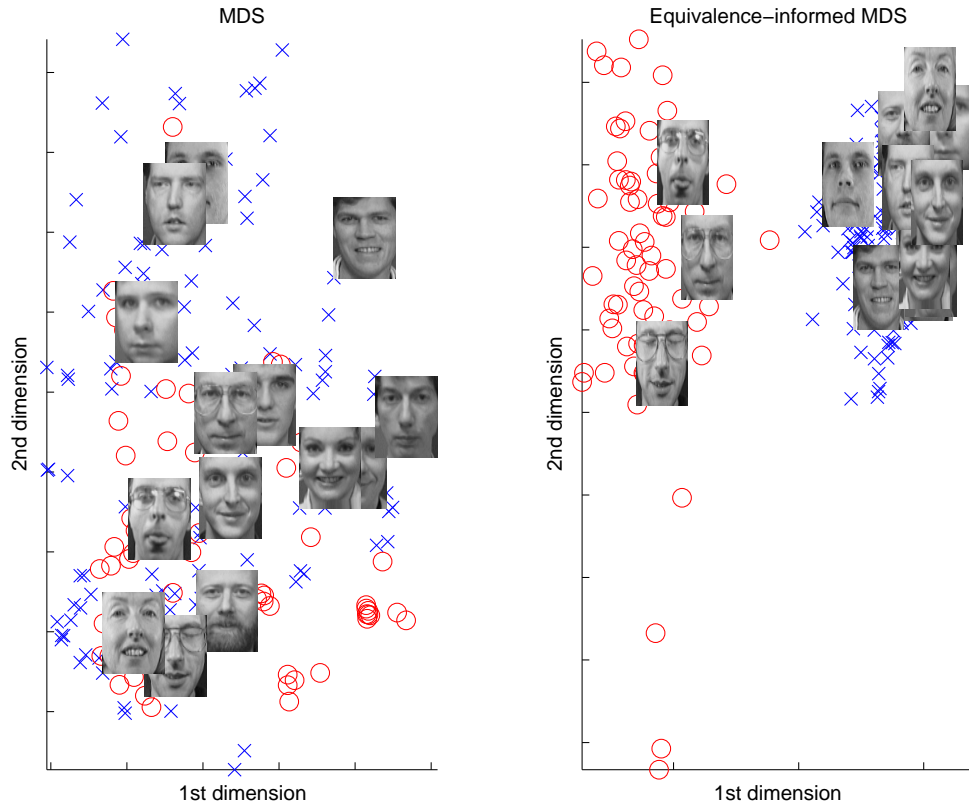


Figure A.1: *MDS and Equivalence-Informed MDS with glasses/no glasses distinction (all class-equivalent pairs)*

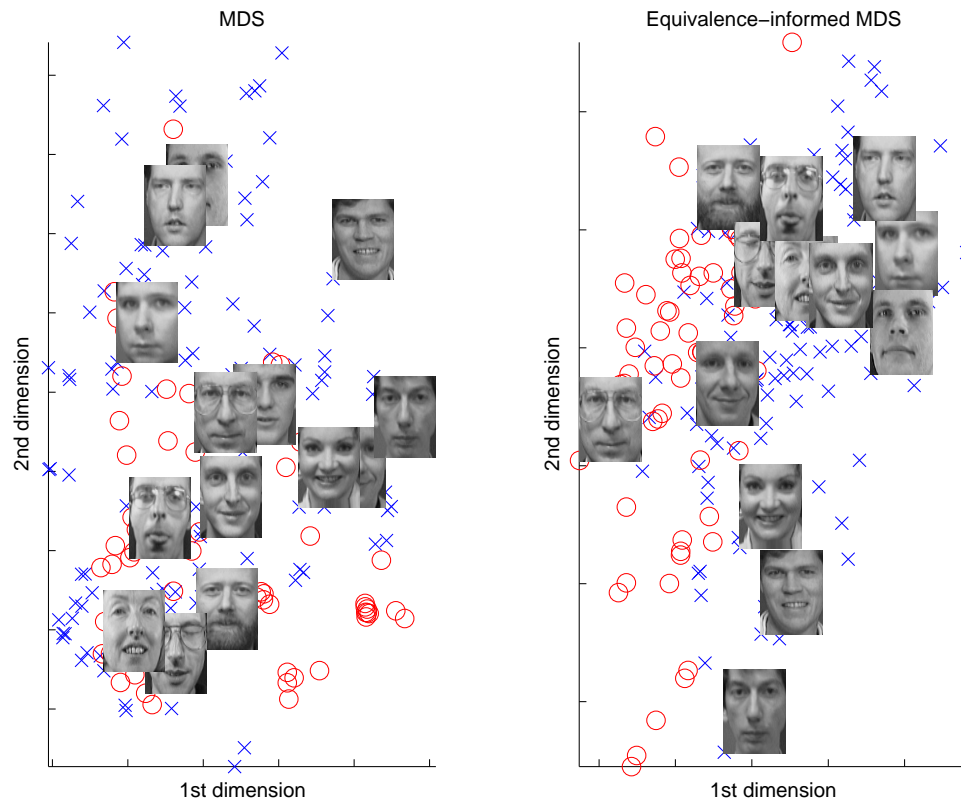


Figure A.2: *MDS and Equivalence-Informed MDS with glasses/no glasses distinction (four class-equivalent pairs, one class-inequivalent pair)*

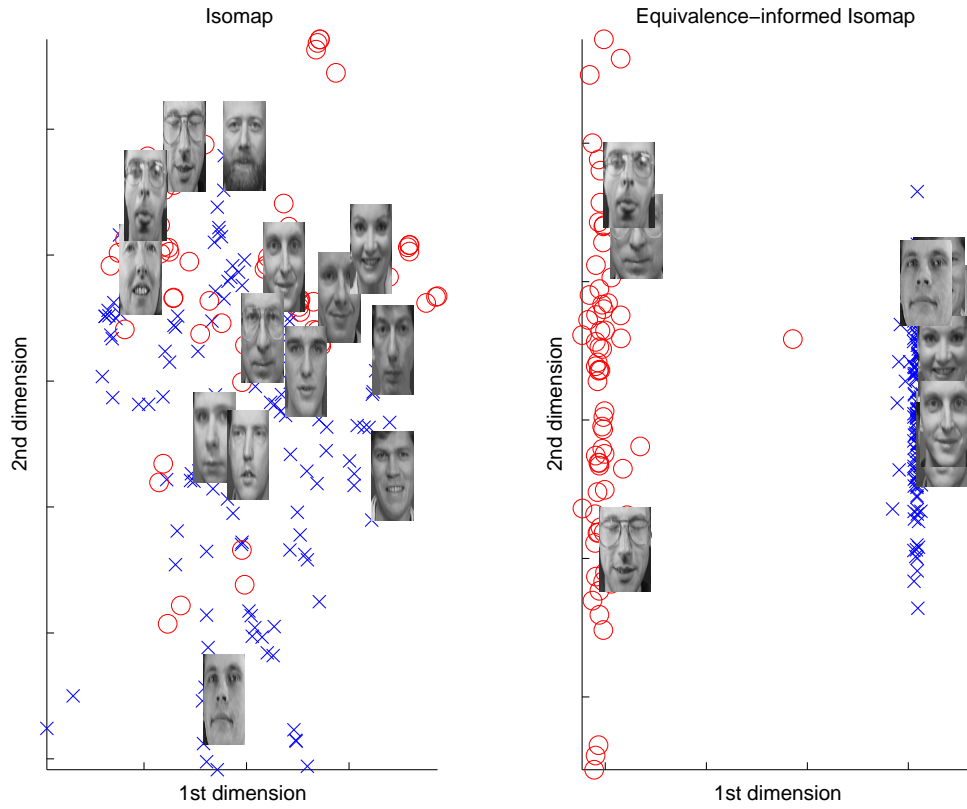


Figure A.3: *Isomap and Equivalence-Informed Isomap with glasses/no glasses distinction (all class-equivalent pairs)*

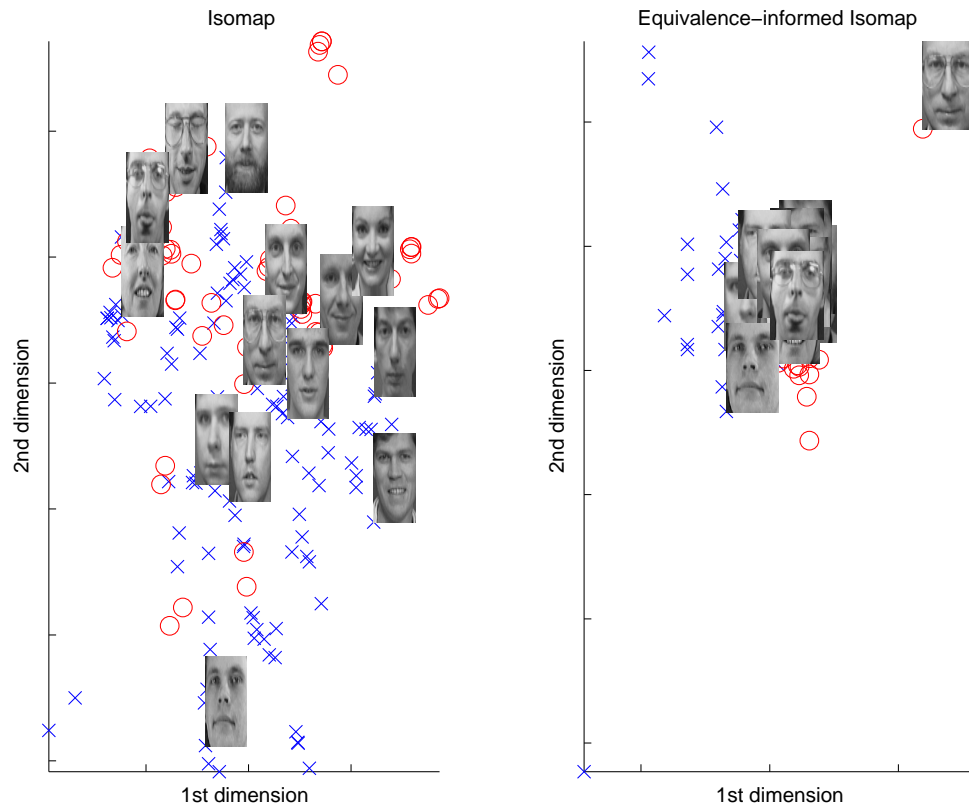


Figure A.4: *Isomap and Equivalence-Informed Isomap with glasses/no glasses distinction (four class-equivalent pairs, one class-inequivalent pair)*

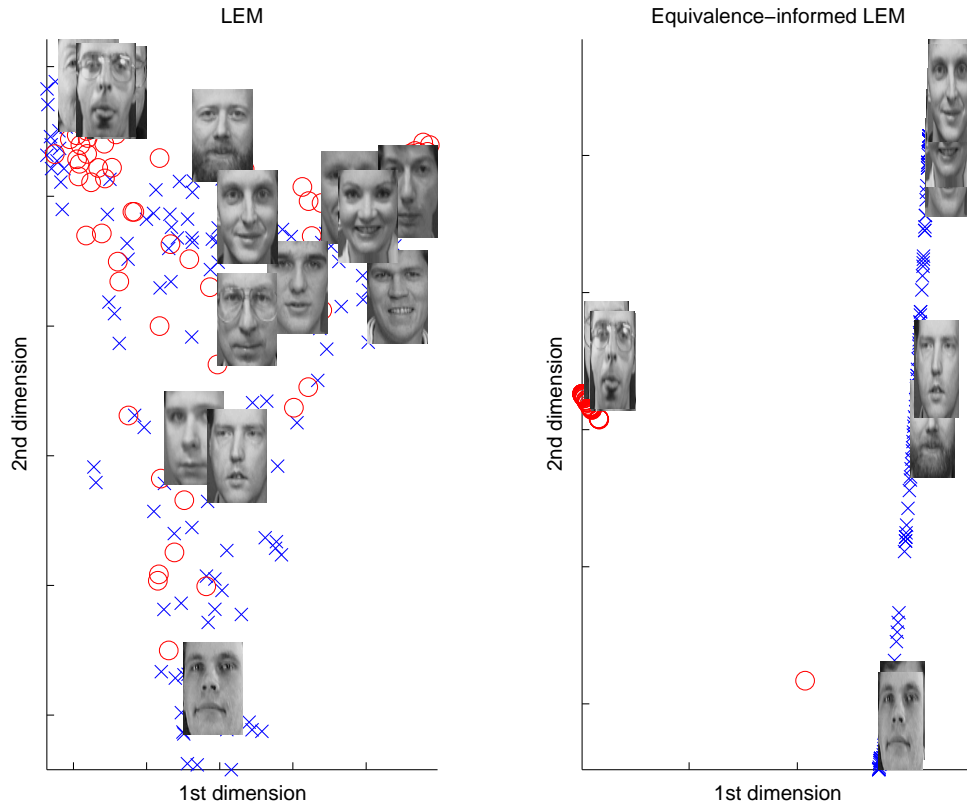


Figure A.5: *LEM and Equivalence-Informed LEM with glasses/no glasses distinction (all class-equivalent pairs)*

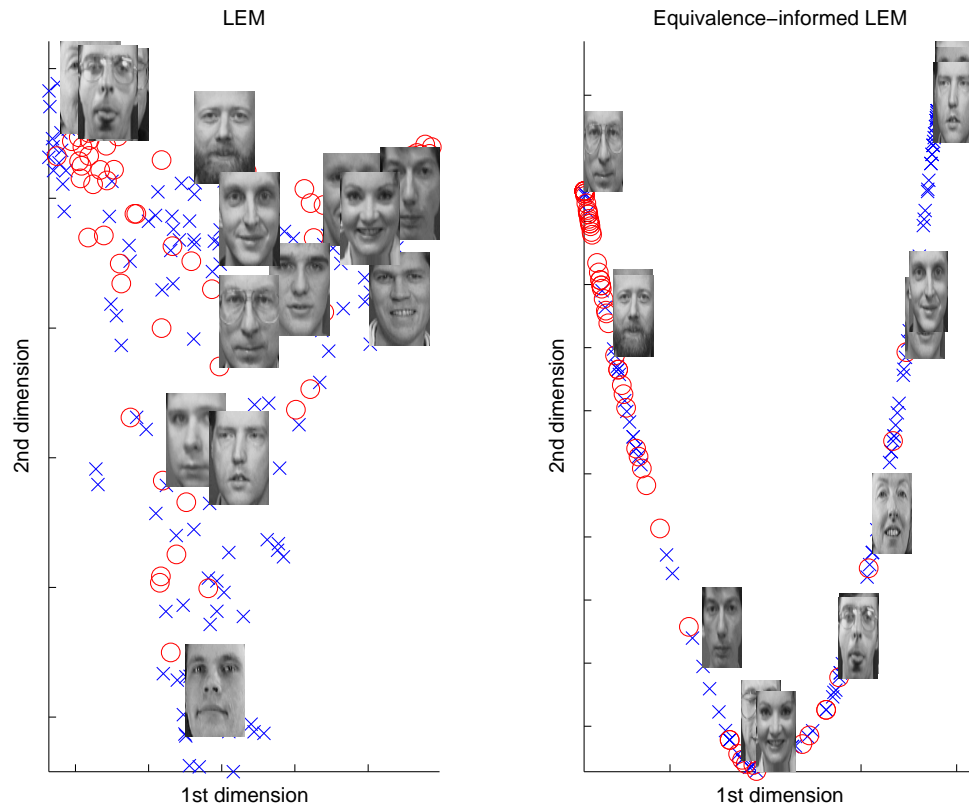


Figure A.6: *LEM and Equivalence-Informed LEM with glasses/no glasses distinction (four class-equivalent pairs, one class-inequivalent pair)*

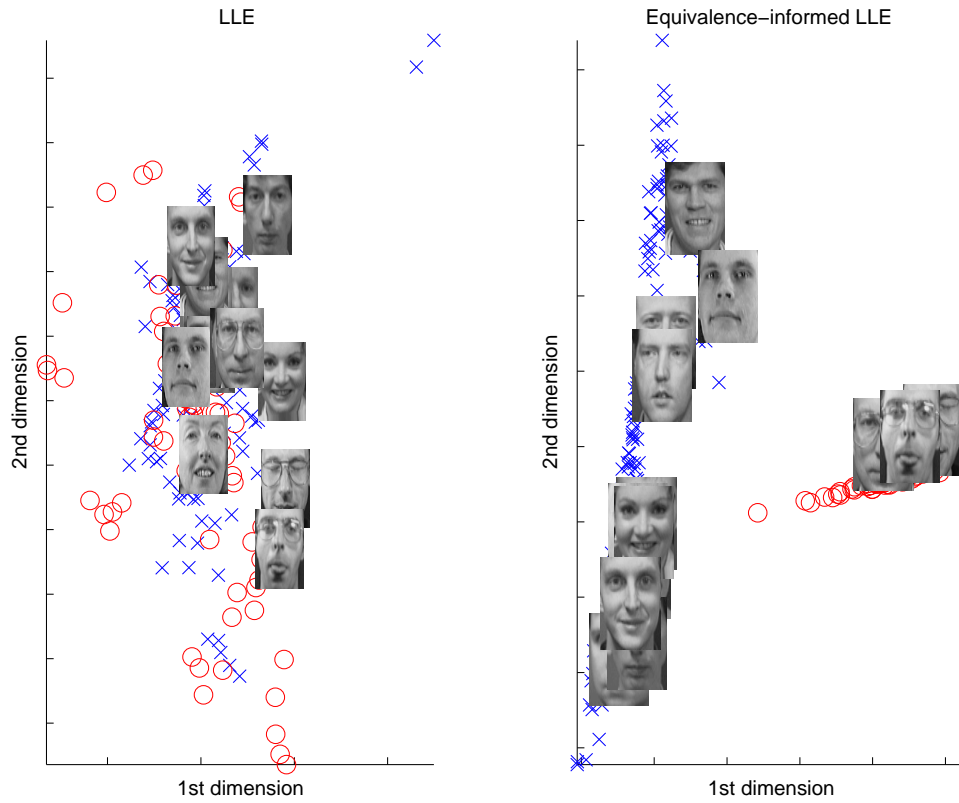


Figure A.7: *LLE and Equivalence-Informed LLE with glasses/no glasses distinction (all class-equivalent pairs)*

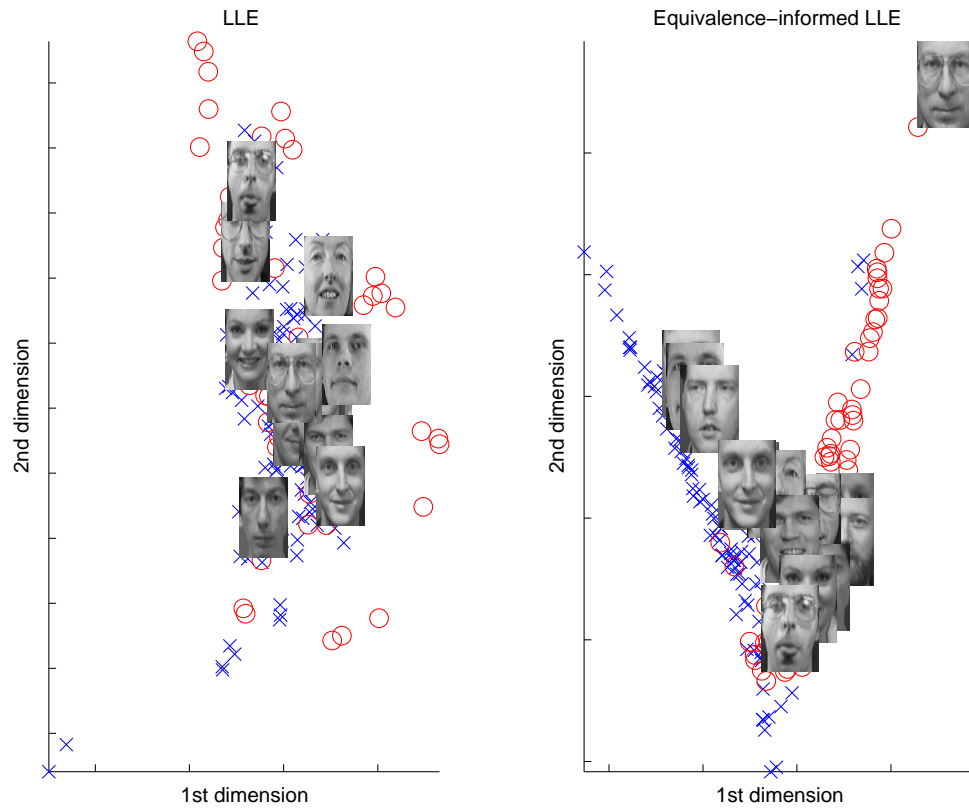


Figure A.8: *LLE and Equivalence-Informed LLE with glasses/no glasses distinction (four class-equivalent pairs, one class-inequivalent pair)*

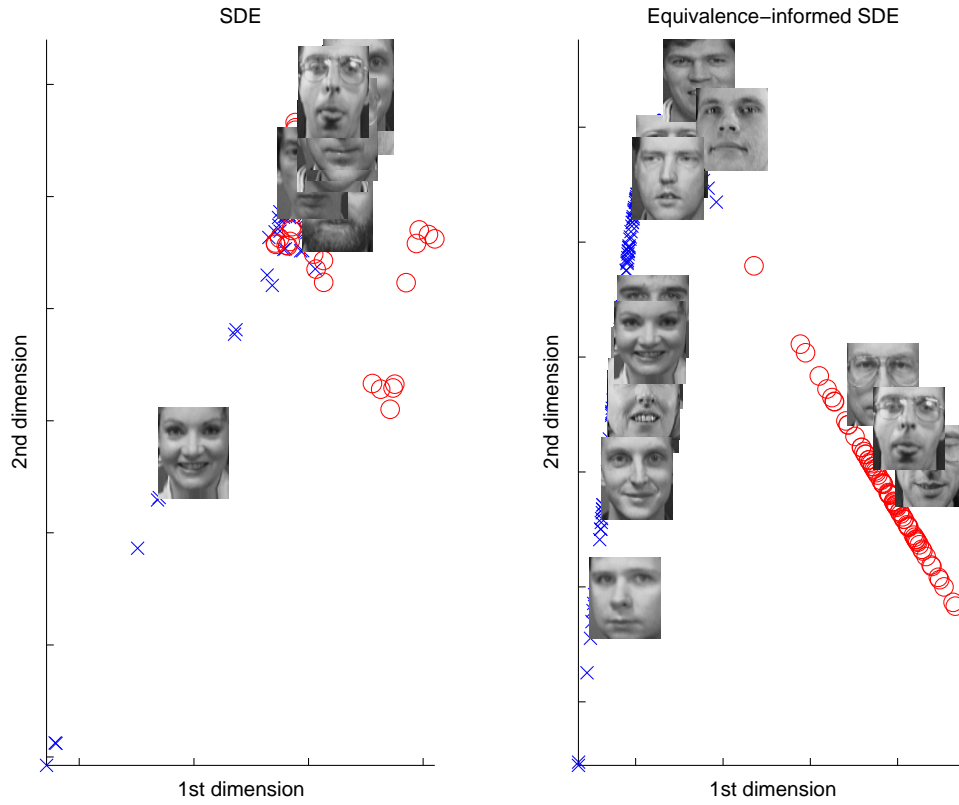


Figure A.9: *SDE and Equivalence-Informed SDE with glasses/no glasses distinction (all class-equivalent pairs)*

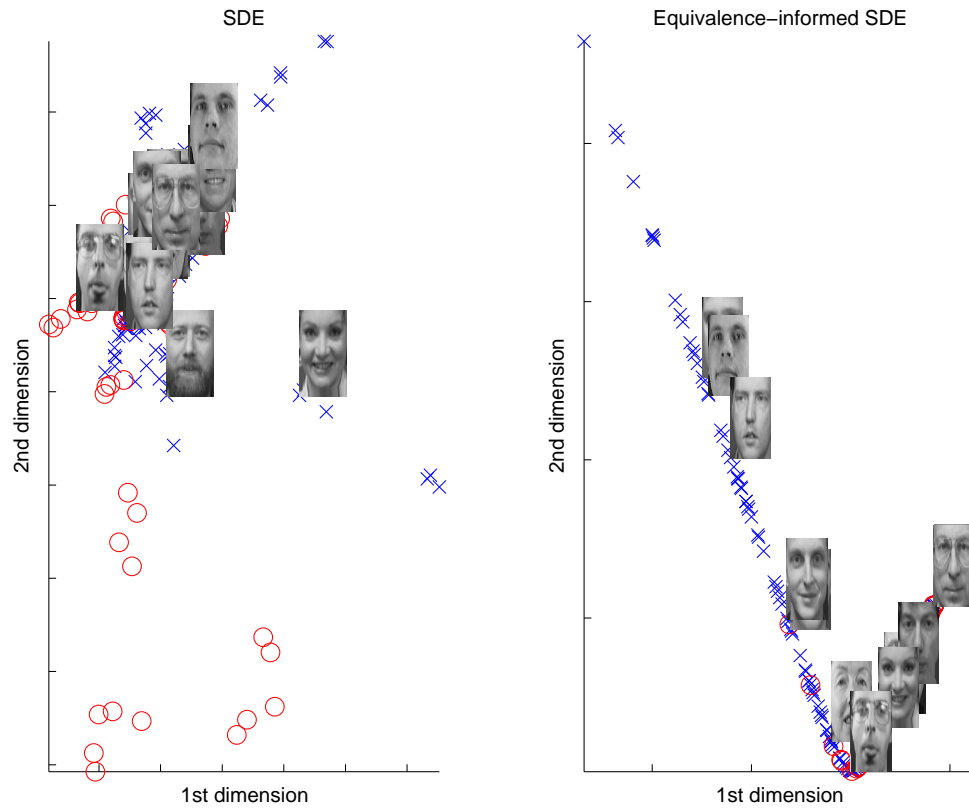


Figure A.10: *SDE and Equivalence-Informed SDE with glasses/no glasses distinction (four class-equivalent pairs, one class-inequivalent pair)*

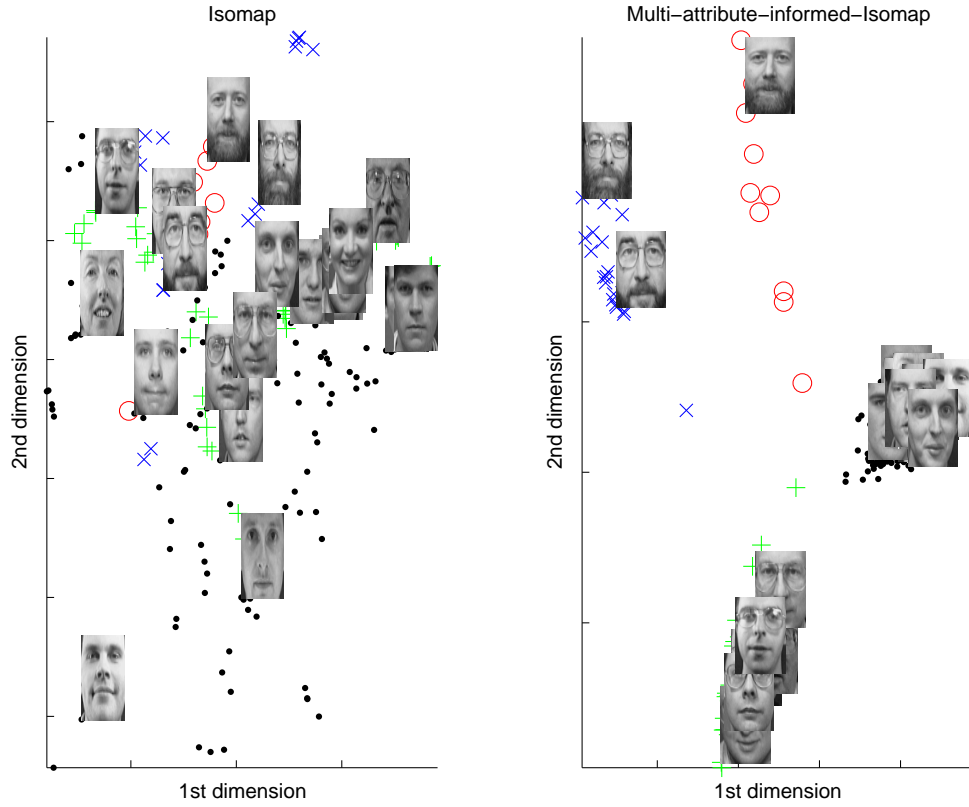


Figure A.11: *Isomap and Multiple-Attribute Equivalence-Informed Isomap with bearded/unbearded and glasses/no glasses distinctions (all class-equivalent pairs)*

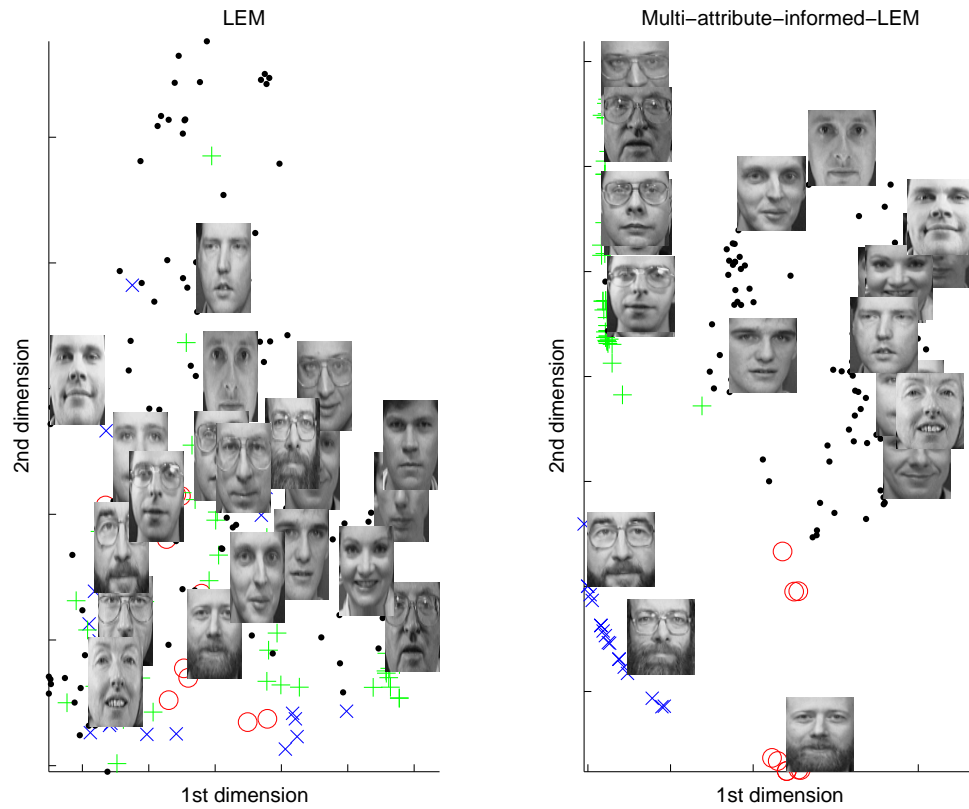


Figure A.12: *LEM and Multiple-Attribute Equivalence-Informed LEM with bearded/unbearded and glasses/no glasses distinctions (all class-equivalent pairs)*

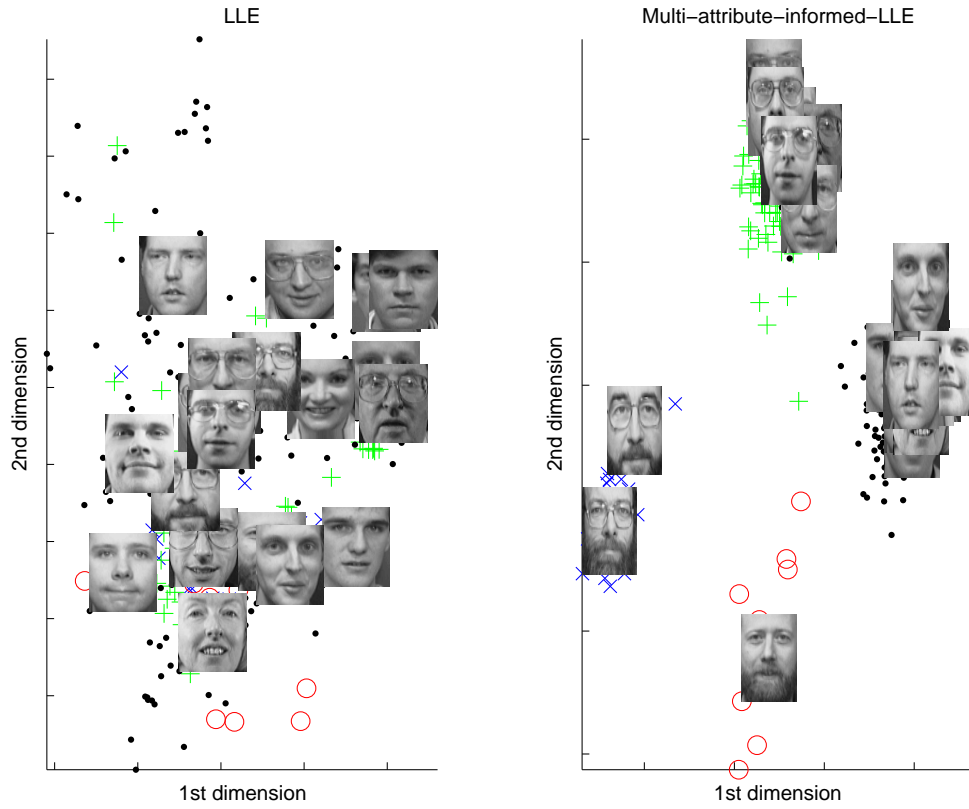


Figure A.13: *LLE and Multiple-Attribute Equivalence-Informed LLE with bearded/unbearded and glasses/no glasses distinctions (all class-equivalent pairs)*

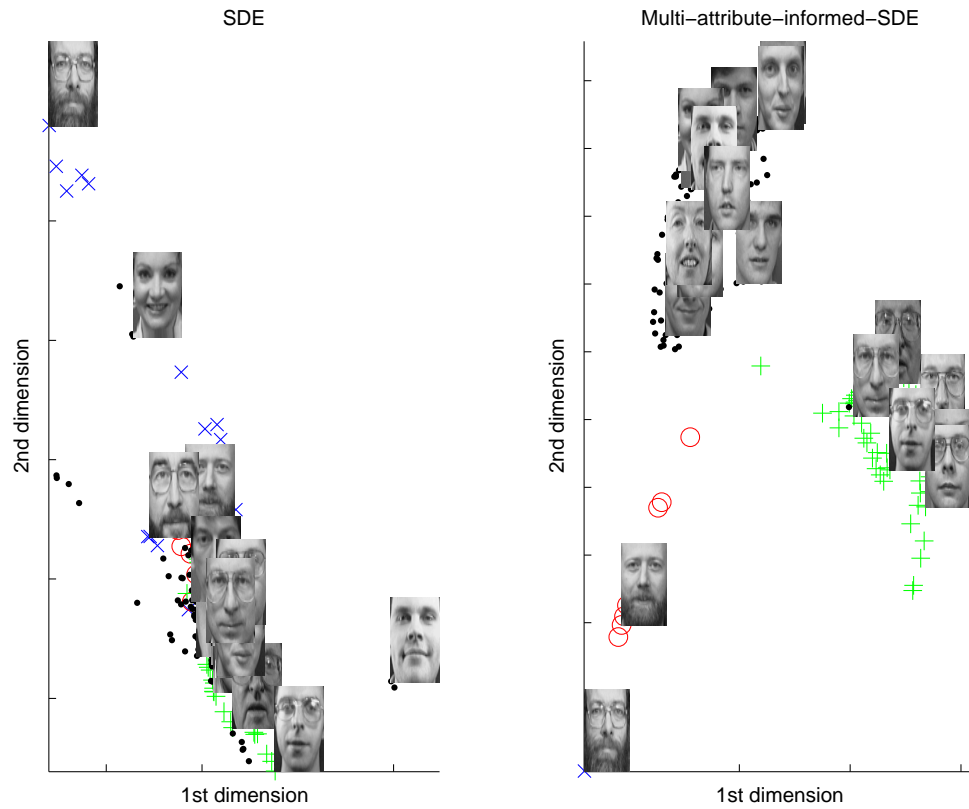


Figure A.14: *SDE and Multiple-Attribute Equivalence-Informed SDE with bearded/unbearded and glasses/no glasses distinctions (all class-equivalent pairs)*

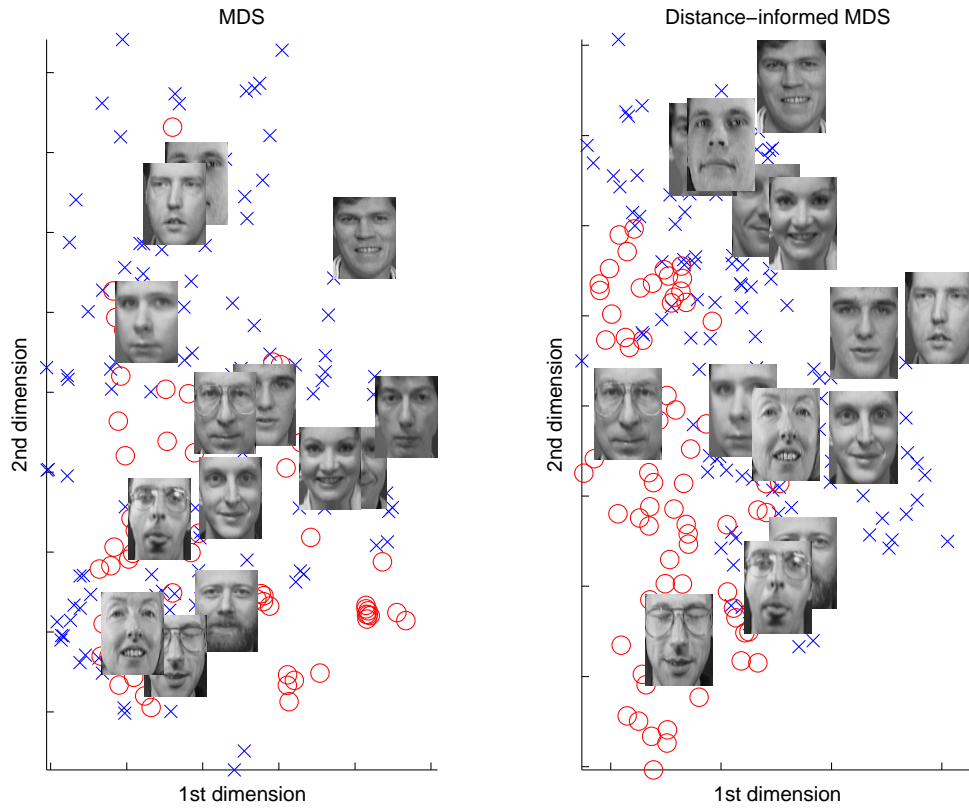


Figure A.15: *MDS and Distance-Informed MDS with glasses/no glasses distinction (30 distances)*

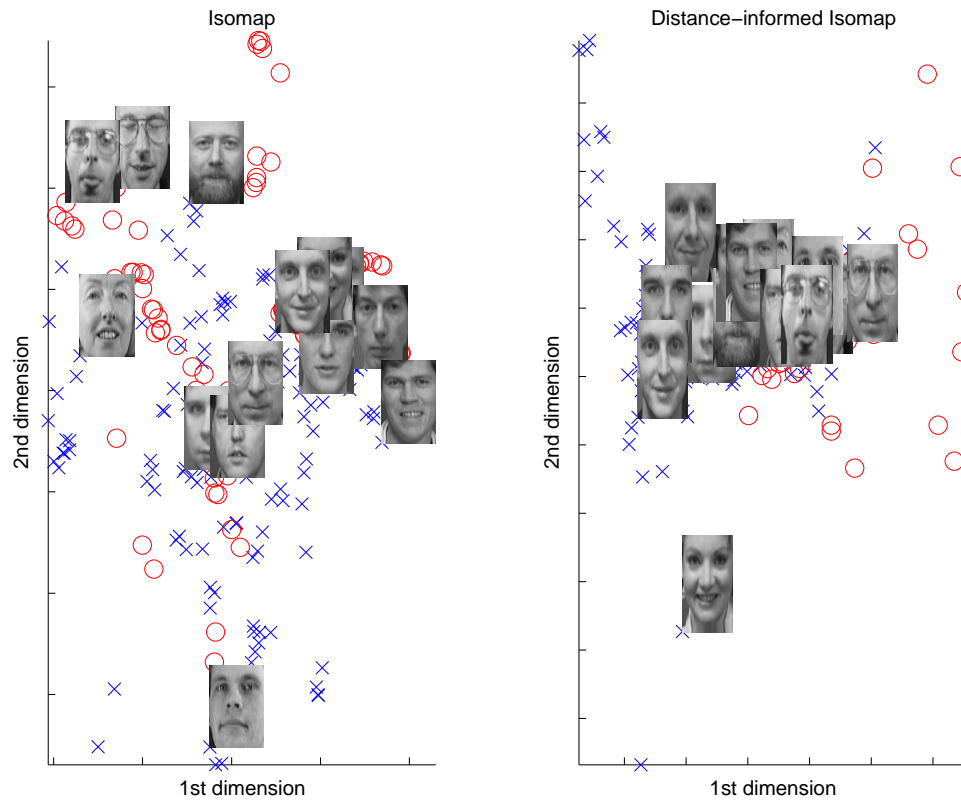


Figure A.16: *Isomap and Distance-Informed Isomap with glasses/no glasses distinction (30 distances)*

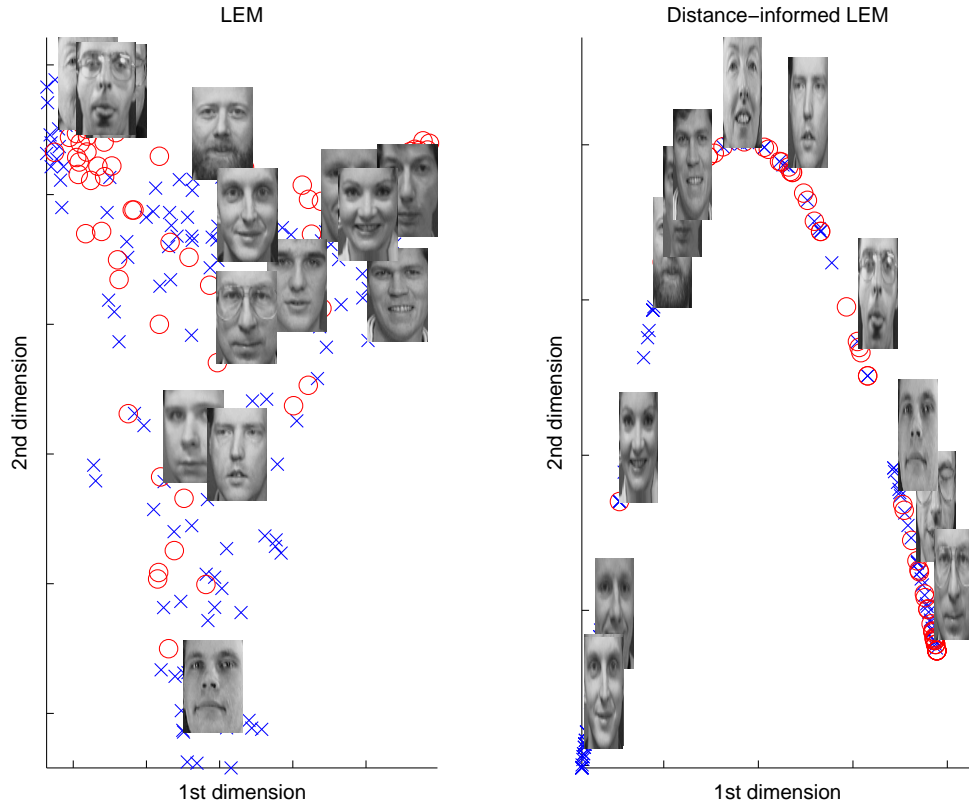


Figure A.17: *LEM and Distance-Informed LEM with glasses/no glasses distinction (30 distances)*

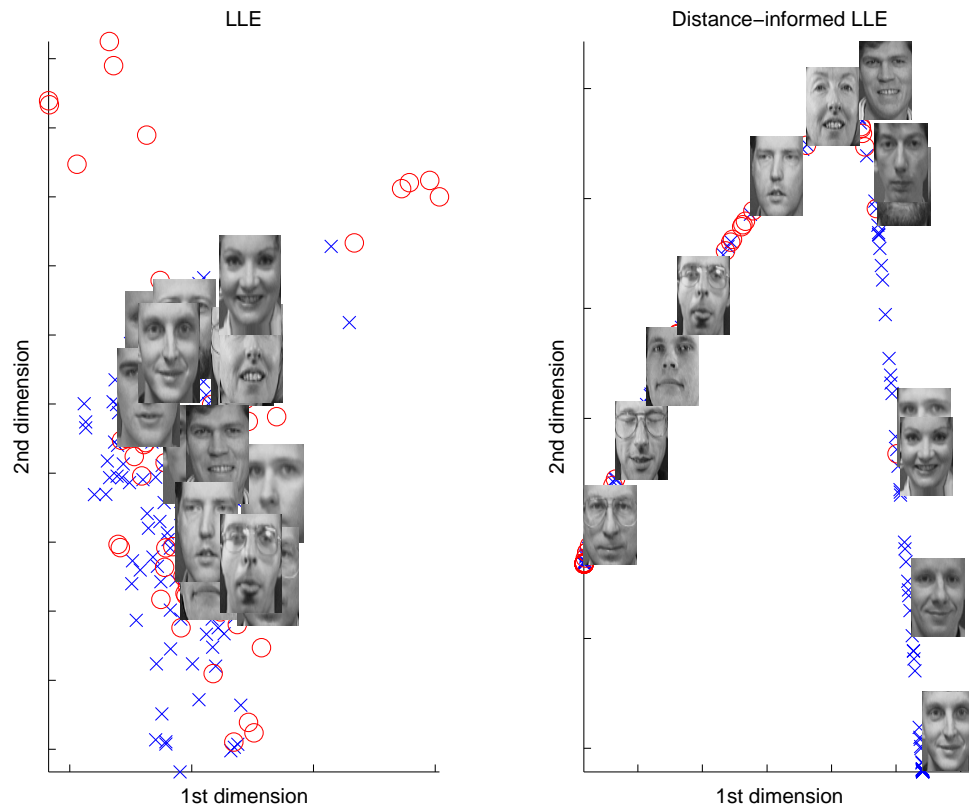


Figure A.18: *LLE and Distance-Informed LLE with glasses/no glasses distinction (30 distances)*

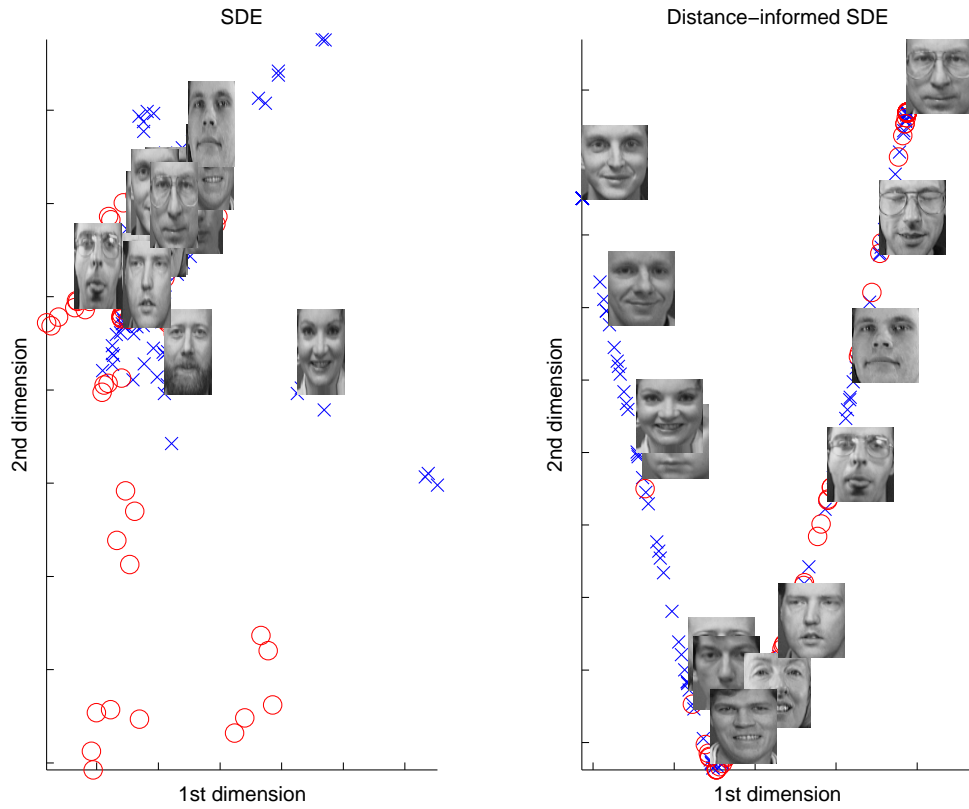


Figure A.19: *SDE and Distance-Informed SDE with glasses/no glasses distinction (30 distances)*

Bibliography

- [1] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.
- [2] Y. Bengio and M. Monperrus. Non-local manifold tangent learning. In *Advances in Neural Information Processing Systems 17*, pages 129–136, 2004.
- [3] M. Black and A. Jepson. EigenTracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, 26(1):63–84, 1998.
- [4] M. Bowling, A. Ghodsi, and D. Wilkinson. Action respecting embedding. In *International Conference on Machine Learning*, 2005.
- [5] M. Bowling, D. Wilkinson, A. Ghodsi, and A. Milstein. Subjective localization with action respecting embedding. In *The International Symposium of Robotics Research*, 2005.

- [6] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, New York, 2004.
- [7] V. Cherkassky and F. Mulier. *Learning from data*. Wiley, New York, 1998.
- [8] D. Cohn. Informed projections. In *Advances in Neural Information Processing Systems 15*, pages 849–856, 2003.
- [9] T. Cox and M. Cox. *Multidimensional Scaling*. Chapman Hall, Boca Raton, 2nd edition, 2001.
- [10] G. Crippen and T. Havel. *Distance geometry and molecular conformation*. Research Studies Press Ltd., Letchworth, 1988.
- [11] B. Frey. *Graphical models for machine learning and digital communication*. MIT Press, Cambridge, Mass, 1998.
- [12] B. Frey and N. Jojic. Transformation-invariant clustering and dimensionality reduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–17, January 2003.
- [13] A. Ghodsi, J. Huang, and D. Schuurmans. Transformation-invariant embedding for image analysis. In *The 8th European Conference on Computer Vision*, 2004.

- [14] A. Ghodsi, J. Huang, F. Southey, and D. Schuurmans. Tangent-corrected embedding. In *International Conference on Computer Vision and Pattern Recognition*, 2005.
- [15] J. Ham, D. Lee, S. Mika, and Schölkopf B. A kernel view of the dimensionality reduction of manifolds. In *International Conference on Machine Learning*, 2004.
- [16] H. Hotelling. Analysis of a complex of statistical variables into components. *J. of Educational Psychology*, 24:417–441, 1933.
- [17] A. Hyvärinen. Survey on independent component analysis. *Neural Computing Surveys*, 2:94–128, 1999.
- [18] I. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1986.
- [19] V. Kumar, A. Grama, A. Gupta, and G. Karypis. *Introduction to parallel computing*. Benjamin, Cummings, 1994.
- [20] R. Memisevic and G. Hinton. Multiple relational embedding. In *Advances in Neural Information Processing Systems 16*, pages 505–512, 2004.
- [21] S. Mika, B. Schölkopf, A. Smola, K.-R. Müller, M. Scholz, and G. Rätsch. Kernel PCA and de-noising in feature spaces. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Proceedings NIPS 11*. MIT Press, 1999.

- [22] K. Pearson. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh and Dublin Philosophical Magazine and Journal of Science*, Sixth Series 2:559–572, 1901.
- [23] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.
- [24] L. Saul and S. Roweis. Think globally, fit locally: Unsupervised learning of nonlinear manifolds. *JMLR*, 2003.
- [25] B. Schölkopf, S. Mika, A. Smola, G. Rätsch, and K.-R. Müller. Kernel PCA pattern reconstruction *via* approximate pre-images. In L. Niklasson, M. Bodén, and T. Ziemke, editors, *Proceedings of the 8th International Conference on Artificial Neural Networks*, pages 147–152, 1998.
- [26] B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, Cambridge, Massachusetts, 2002.
- [27] P. Simard, Y. LeCun, and J. Denker. Efficient pattern recognition using a new transformation distance. In *Advances in Neural Information Processing Systems 5*, pages 50–58, 1993.

- [28] J. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optim. Methods Softw.*, 11/12(1-4):625–653, 1999.
- [29] R. Rivest T. Cormen, C. Leiserson and C. Stein. *Introduction to algorithms*. MIT Press, Cambridge, Massachusetts, 2001.
- [30] J. Friedman T. Hastie, R. Tibshirani. *The elements of statistical learning*. Springer, New York, 2002.
- [31] J. Tenenbaum. Mapping a manifold of perceptual observations. In *Advances in Neural Information Processing Systems 10*, pages 682–687, 1998.
- [32] J. Tenenbaum, V. de Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.
- [33] J. B. Tenenbaum and W. T. Freeman. Separating style and content with bilinear models. *Neural Computation*, 12(6), 2000.
- [34] K. Weinberger and L. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *Proceedings of the International Conference on Machine Learning*, pages 839–846, 2004.

- [35] K. Weinberger and L. Saul. Unsupervised learning of image manifolds by semidefinite programming. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 988–995, 2004.
- [36] D. Wilkinson, M. Bowling, and A. Ghodsi. Learning subjective representations. In *The 19th International Joint Conference on Artificial Intelligence*, 2005.
- [37] C. K. I. Williams. On a connection between kernel PCA and metric multidimensional scaling. *Machine Learning*, 46(1-3):11–19, 2002.
- [38] E. Xing, A. Ng, M. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. In *Advances in Neural Information Processing Systems 15*, pages 505–512, 2003.
- [39] P. Vincent Y. Bengio and J.-F. Païement. Learning eigenfunctions of similarity: Linking spectral clustering and kernel pca. Technical Report 1232, Universite de Montreal, 2003.